Privacy-Preserving
Distributed Information Sharing

Lea Kissner

CMU-CS-06-149
July 2006

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Dawn Song, Chair
Manuel Blum
Dan Boneh, Stanford University
Benny Pinkas, Haifa University
Michael Reiter

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

This research was sponsored by the US Army Research Office under contract no. DAAD19-02-1-0389 and the National Science Foundation under subcontract no. SA4896-10808PG.

The views and conclusions contained herein are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government, or any other governmental, commercial or legal entity.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **JUL 2006**	2. REPORT TYPE	3. DATES COVERED **00-07-2006 to 00-07-2006**

4. TITLE AND SUBTITLE **Privacy-Preserving Distributed Information Sharing**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **94**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Keywords: privacy, distributed information sharing, multiset operations, hot-item identification

For Chris, whose support, kindness, and reminders
to eat regularly have made this work possible.

Abstract

In many important applications, a collection of mutually distrustful parties must share information, without compromising their privacy. Currently, these applications are often performed by using some form of a trusted third party (TTP); this TTP receives all players' inputs, computes the desired function, and returns the result. However, the level of trust that must be placed in such a TTP is often inadvisable, undesirable, or even illegal. In order to make many applications practical and secure, we must remove the TTP, replacing it with efficient protocols for privacy-preserving distributed information sharing. Thus, in this thesis we explore techniques for privacy-preserving distributed information sharing that are efficient, secure, and applicable to many situations.

As an example of privacy-preserving information sharing, we propose efficient techniques for privacy-preserving operations on multisets. By building a framework of multiset operations, employing the mathematical properties of polynomials, we design efficient, secure, and composable methods to enable privacy-preserving computation of the *union, intersection,* and *element reduction* operations. We apply these techniques to a wide range of practical problems, including the Set-Intersection, Over-Threshold Set-Union, Cardinality Set-Intersection, and Threshold Set-Union problems. Additionally, we address the problem of determining Subset relations, and even use our techniques to evaluate CNF boolean formulae.

We then examine the problem of hot item identification and publication, a problem closely related to Over-Threshold Set-Union. Many applications of this problem require greater efficiency and robustness than any previously-designed secure protocols for this problem. In order to achieve sufficiently efficient protocols for these problems, we define two new privacy properties: *owner privacy* and *data privacy.* Protocols that achieve these properties protect the privacy of each player's personal input set, as well as protecting information about the players' collective inputs. By designing our protocols to achieve owner and data privacy, we are able to significantly increase efficiency over our privacy-preserving set operations, while still protecting the privacy of participants. In addition, our protocols are extremely flexible - nodes can join and leave at any time.

Acknowledgements

I would like to thank my (impressively patient) family and friends[1] for supporting me through the highs and lows of my graduate school experience. I would like to thank my dance shoes for supporting my arches and salsa habit.

I'd like to thank Dawn Song, my advisor. I've learned so much from her over the last four years, and this thesis could not have been written without her guidance. My thesis committee, Manuel Blum, Dan Boneh, Benny Pinkas and Mike Reiter, provided me with perspective, many valuable insights and comments on this work which improved it in innumerable ways.

I'd like to thank **Catherine Copetas** and **Sharon Burks** in a **large** font, and not just because it's a required part of the thesis. I have always appreciated the wisdom they've shared with me, as well as their scarily efficient ability to get things done when asked nicely.

The set operations work in this thesis was performed with my advisor, Dawn Song. The hot item work in this thesis was performed with Hyang-Ah Kim, Dawn Song, Oren Dobzinski and Anat Talmy.

I would like to thank David Molnar, Christopher Colohan, Lujo Bauer, Nikita Borisov, and Alina Oprea for their invaluable comments on the hot item identification work in this thesis. I would also like to extend my thanks to Dan Boneh, Benny Pinkas, David Molnar, and Alexandre Evfimievski for their invaluable help and comments on the content and presentation of the multiset operations work in this thesis. My thanks also go out to Luis von Ahn, Lujo Bauer, David Brumley, Bryan Parno, Alina Oprea, Mike Reiter, and the many anonymous reviewers who reviewed my papers and provided valuable feedback which enhanced the work which you read here. Finally, I'd like to thank David Molnar again for his help in proofreading my thesis.

[1] Note that this thesis is about preserving privacy. Thus, I have not listed my family and friends to preserve their privacy and give them plausible deniability. You know who you are.

Contents

List of Figures

Chapter 1

Introduction

As computer system capacity has increased, organizations and individuals have collected greater and greater amounts of data. Developments in data mining have increased the utility of this data; a data holder can now discover hidden trends and cheating customers. However, a great deal of useful data cannot be computed from one party's data alone. When multiple parties can compare and share data, they can greatly increase the utility of their data. For example, multiple pharmacies might compare their records to detect people filling a prescription multiple times. Subtle trends can leave fingerprints in many data sets, hiding them from detection unless multiple data sets are examined. The early stages of a disease epidemic might cause an increased rate of absenteeism from work and school, higher sales of certain over-the-counter medications, and many other small traces. Each of these traces, on its own, might not be enough to detect an epidemic, as they fluctuate based on many factors. By combining many data sets, we can detect increasingly more subtle trends.

Data does not exist in a void, however. To detect trends that involve medical data, we must use medical data collected from individuals. This data is generally considered to be private; many countries have strict regulations that control the use of medical and other personal information. Many other data sets are collected by companies who are concerned about preserving the proprietary value of their data. Government data often has both privacy and security restrictions associated with its use. Thus, we often cannot simply combine data sets held by multiple parties to compute functions over the combined data. In the real world, parties often resort to use of a trusted third party, who computes a fixed function on all parties' private inputs, or forgo the application altogether. This unconditional trust is fraught with security risks; the trusted party may be dishonest or compromised, as it is an attractive target. The problem of *privacy-preserving distributed information sharing* is to allow parties with private data sets to compute these joint functions without use of a trusted party, and thus achieve many of the benefits obtained from combining the data sets without undesirably revealing private data.

Protocols for privacy-preserving distributed information sharing must also be designed around a number of practical concerns. Many data sets are extremely large; protocols that operate on large data sets must be efficient in order to operate in the real world. In addition, we must be concerned with robustness. Adversaries may attempt to manipulate the protocol to learn private information or change the results. Some adversaries may even manipulate the network, causing some players to become disconnected from others. In some problem scenarios, such as those concerning detecting and defending against network attacks, robustness against an unreliable network is paramount.

In this thesis, we examine several specific problems that fall under the heading of privacy-preserving distributed information sharing. We design efficient protocols for these problems, proving their security and correctness in the presence of attackers. The first set of problems we examine are those related to privacy-preserving set and multiset operations. We then examine in depth the problem of hot item identification, including its need for extreme efficiency and robustness.

1.1 Privacy-Preserving Set and Multiset Operations

We design efficient *privacy-preserving* techniques and protocols for computation over multisets by mutually distrustful parties: no party learns more information about other parties' private input sets than what can be deduced from the result of the computation.

For example, to determine which airline passengers appear on a 'do-not-fly' list, the airline must perform a set-intersection operation between its private passenger list and the government's list. This is an example of the Set-Intersection problem. If a social services organization needs to determine the list of people on welfare who have cancer, the union of each hospital's lists of cancer patients must be calculated (but not revealed), then an intersection operation between the unrevealed list of cancer patients and the welfare rolls must be performed. This problem may be efficiently solved by composition of our private union and set-intersection techniques.

Another example of the use of these techniques is in privacy-preserving distributed network monitoring. In this scenario, each node monitors anomalous local traffic, and a distributed group of nodes collectively identify popular anomalous behaviors: behaviors that are identified by at least a threshold t number of monitors. This is an example of the Over-Threshold Set-Union problem.

In this thesis, we propose efficient techniques for privacy-preserving operations on multisets. By building a framework of multiset operations using polynomial representations and employing the mathematical properties of polynomials, we design efficient methods to enable privacy-preserving computation of the *union, intersection, and element reduction*[1] multiset operations.

An important feature of our privacy-preserving multiset operations is that they can be composed, and thus enable a wide range of applications. To demonstrate the power of our techniques, we apply our operations to solve specific problems, including Set-Intersection, Cardinality Set-Intersection, Over-Threshold Set-Union, and Threshold Set-Union, as well as determining the Subset relation. Furthermore, we show that our techniques can be used to efficiently compute the output of any function over multisets expressed in the following grammar, where s represents any set held by some player and $d \geq 1$:

$$\Upsilon ::= s \mid \mathrm{Rd}_d(\Upsilon) \mid \Upsilon \cap \Upsilon \mid s \cup \Upsilon \mid \Upsilon \cup s$$

Note that any monotonic function over multisets[2] can be expressed using our grammar, showing that our techniques have truly general applicability. Finally, we show that our techniques are applicable even outside the realm of set computation. As an example, we describe how to utilize our techniques to efficiently and privately evaluate CNF boolean functions.

Our protocols are more efficient than the results obtained from previous work. General multiparty computation is the best previous result for most of the multiset computation problems

[1] The *element reduction* by d, $\mathrm{Rd}_d(A)$, of a multiset A is the multiset composed of the elements of A such that for every element a that appears in A at least $d' > d$ times, a is included $d' - d$ times in $\mathrm{Rd}_d(A)$.

[2] Any function computed with only intersection and union, without use of an inverse operation.

we address in this thesis. Only the private Set-Intersection problem and two-party Cardinality Set-Intersection problem have been previously studied [1, 23]. However, previous work only provides protocols for 3-or-more-party Set-Intersection secure only against honest-but-curious players; it is not obvious how to extend this work to achieve security against malicious players. Also, previous work focuses on achieving results for the Set-Intersection problem in isolation – these techniques cannot be used to compose set operations. In contrast, we provide efficient solutions for private multi-party Set-Intersection secure against malicious players, and our multiset intersection operator can be easily composed with other operations to enable a wide range of efficient private computation over multisets. We compare the communication complexity of our protocols with previous work and solutions based on general multiparty communication in Table 4. Note that the techniques utilized to create the circuits for the general solution are both complex and incur very large constants, on top of the constants inherent in the use of general multiparty computation [2]; we thus achieve greater practical efficiency, as well as asymptotic efficiency.

Our protocols are provably secure in the PPT-bounded adversary model. We consider both standard adversary models: honest-but-curious adversaries (HBC) and malicious adversaries. For protocols secure in the HBC model, we prove that the information learned by any coalition of honest-but-curious players is indistinguishable from the information learned in the ideal model, where a trusted third party (TTP) calculates the function. For protocols secure in the malicious model, we provide simulation proofs showing that for any strategy followed by a malicious coalition Γ in the real protocol, there is a translated strategy they could follow in the ideal model, such that, to Γ, the real execution is computationally indistinguishable from ideal execution.

1.2 Privacy-Preserving Distributed Hot Item Identification and Publication

In this thesis, we consider a scenario in which a group of distributed nodes, each holding its local data set, would like to collectively identify commonly occurring items. More formally, a "commonly occurring item" is one that appears in at least a threshold number of nodes' local data sets. We call such items *hot items*. The problem of identifying and publishing these hot items is closely related to the Over-Threshold Set-Union problem we examine in Chapter 4. Distributed identification of hot items, while preserving privacy, is important for many applications.

For example, in distributed network monitoring, each participant monitors its local traffic and the participants collectively need to identify common offenders (IP addresses that are flagged as malicious by multiple sites) and common alerts (events that are flagged as anomalous or malicious by multiple sites). Identifying common offenders and alerts is important to enable defenses against wide-spread attacks as well as reduce the false positive rate; an offender or alert reported by multiple sites is more likely to be truly malicious.

Many more applications use hot item identification for statistics gathering. In computer troubleshooting, common configurations among unbroken computers can be used to identify configuration errors and suggest fixes for troubled computers [25, 30, 55]. In a distributed content delivery network (CDN), distributed identification of hot pages (web pages that are commonly requested at different sites) is important for making effective caching decisions; hot pages should have higher priority when caching [10].

In each of these applications, it is crucial to identify hot items held by distributed players. At the same time, each local data set, be it local network traffic, a computer's configurations, or a site's web surfing traffic, often contains personal or security-critical data; thus, we need effective methods to identify hot items without revealing information about the non-hot items. Moreover, in many distributed applications, some participants may not be trustworthy, and may even be malicious. Thus, we need to design effective methods to enable distributed privacy-preserving hot item identification in the presence of malicious participants. Additionally, as players may join and leave the network frequently, we must construct protocols that do not require global knowledge of the network.

Efficient, secure, and privacy-preserving hot item identification and publication is a challenging problem; previous solutions are largely insufficient. One approach is for every participant to send his data to a trusted central authority, who identifies and announces the hot items. In this approach, all security and privacy relies on the central authority. This level of trust is unacceptable for many situations. Even if the authority is trusted, an attacker may compromise the authority and players' privacy with it. Several other problems inherent in centralization are discussed in [37].

Another approach is to use a partially homomorphic cryptosystems, such as a distributed version of Paillier [21, 22, 45], or a secure multi-party computation scheme to compute the frequencies of each item or to identify hot items directly. However, these methods have significant computation and communication overhead, including zero-knowledge proofs, which is often prohibitive for large scale applications. Key management for maintaining shared keys in the presence of malicious parties and players that join and leave the network may add non-trivial overhead and complexity.

All previous approaches (except our Over-Threshold Set-Union protocols of Chapter 4) of which we are aware propose heuristic solutions [37]; some even require additional trust assumptions such as "friends" [30, 55]. They do not preserve several forms of privacy that we believe are important (see Section 5.1), do not give rigorous analysis, and cannot prevent malicious participants from changing the result arbitrarily.

In this thesis, we propose new techniques for efficient, secure, and privacy-preserving distributed hot item identification and publication. To avoid counting the occurrences of each hot item separately, we utilize a probabilistic filtering technique, allowing both efficiency and privacy. Each player constructs a local filter, which is then combined with those of other players to create a global filter. In the process of combination, we utilize an approximate counting technique which is both efficient and secure against undue interference by malicious parties.

Protocols for hot item identification and publication that achieve standard cryptographic definitions of privacy [28] are too inefficient for many applications, including our protocols of Chapter 4. We design protocols that enable these demanding applications by trading a certain degree of privacy for greater efficiency; as a result, our protocols are comparably asymptotically efficient to approaches that do not protect the privacy of participants, as described in Section 5.4.4. We also construct one-show tags to prevent malicious players from tampering with the identification of hot items. Our protocols scale extremely well when increasing the number of players. If the hot-item threshold is proportional to the number of players, then the bandwidth used per node is essentially constant as the number of players increases, as is optimal. (See Section 5.4.4.)

Elements of honest players' private input sets are protected by *data privacy.* This property, which we rigorously define, is weaker than a standard notion of cryptographic privacy [28], but can be achieved more efficiently. Essentially, the data privacy property states that non-hot items are hidden in a crowd of indistinguishable elements, and that the more rare an item is, the less information is revealed about it. Players who publish their hot items are protected by the property *owner privacy*, which we rigorously define. Players may choose between *correlated owner privacy*, in which published elements cannot be associated with the publishing player, and *uncorrelated owner privacy*, in which we enforce the additional guarantee that no player can distinguish whether two items have appeared in the same private input set.

Our protocol prevents a group of malicious players from influencing the identification and publication of hot items: no group of malicious players may cause any element to be identified as a hot item with higher probability than if it simply appeared in the malicious players' private input sets.

Unlike previous work, our protocols are extremely flexible in situations in which untrusted clients often join and leave the network. As we require no threshold cryptography, secure multi-party computation, or global knowledge of the network, no consistent set of players is needed to execute a protocol. No player need trust any other.

We prove bounds on the probability of correctness of our protocols, as well as for data- and owner-privacy. The approach we introduce in this thesis is applicable to many situations. Our protocols are the most efficient hot item identification and publication protocols of which we are aware that achieve the properties of data and owner privacy.

1.3 Thesis Outline

We begin by discussing related work in Chapter 2 and cryptographic and mathematical preliminaries in Chapter 3. We then introduce our techniques and protocols for privacy-preserving set and multiset computation in Chapter 4. In Chapter 5, we introduce our protocols for privacy-preserving distributed hot item identification and publication. We conclude in Chapter 6. We include additional proofs and information about our results in Chapter 5 in Appendix B.

Chapter 2

Related Work

In this chapter, we consider previous work related to our privacy-preserving multiset operations and hot-item identification. We then discuss the distinctions between our results and previous work.

2.1 Works Related to Multiset Operations

Most of the privacy-preserving multiset operations and functions we address in this thesis (Chapter 4) have no better result in previous work than through general multiparty computation. General two-party computation was introduced by Yao [56], and general computation for multiple parties was introduced in [5]. In general multiparty computation, the players share the values of each input, and cooperatively evaluate the circuit. For each multiplication gate, the players must cooperate to securely multiply their inputs and re-share the result, requiring $O(n)$ communication for honest-but-curious players and $O(n^2)$ communication for malicious players [28]. Recent results that allow non-interactive private multiplication of shares [16] do not extend to our adversary model (see Section 3.1), in which any $c < n$ players may collude. Our results are more efficient than the general MPC approach; we compare communication complexity in Table 4.

One privacy-preserving function that has been considered in both our results and previous work is set intersection. Rakesh Agrawal and Alexandre Evfimievski and Ramakrishnan Srikant [1] and Freedman, Nissim, and Pinkas (FNP) [23] proposed protocols for problems related to two party Set-Intersection. FNP proposed protocols for multiparty set intersection (secure only against honest-but-curious players) and two-party cardinality set intersection as well. FNP's results are based on the representation of sets as roots of a polynomial [23]. Their work does not utilize properties of polynomials beyond evaluation at given points. In Chapter 4 of this thesis, we explore the power of polynomial representation of multisets, using operations on polynomials to obtain three composable privacy-preserving *multiset* operations. We give a more detailed comparison of our Set-Intersection protocol with FNP in Table 4.

In addition to previous work on privacy-preserving set intersection, researchers have designed protocols for privacy-preserving computation of several related functions. For example, private equality testing is the problem of set-intersection for the case in which the size of the private input sets is 1. Protocols for this problem are proposed in [19, 38, 43], and fairness is added in [8]. Another related problem is in testing the disjointness of private input sets [34]; a restricted version of the Cardinality Set-Intersection problem. We do not enumerate the works of privacy-preserving computation of other functions here, as they address drastically different problems and cannot be applied to our setting.

2.2 Works Related to Hot-Item Identification

In essence, hot-item identification is a small variation on the problem of Over-Threshold Set-Union. We address the Over-Threshold Set-Union problem in Chapter 4, with previous work in a cryptographically secure setting considered in Section 2.1. In hot-item identification, the players in the protocol approximate the desired results, and give up a certain measure of privacy in exchange for increased efficiency and robustness. For example, our Over-Threshold Set-Union protocol requires the players to share a decryption key and perform joint decryption. We do not believe that such an assumption is tenable in all situations.

Several applications of privacy-preserving hot item identification and publishing have been considered in previous work. Certain privacy-breaching attacks against distributed network monitoring nodes were described in [37]. They did not, however, give a concrete definition of security for their attempts to defeat such attacks, and their techniques require trusted central servers. Additionally, in many cases, significant breaches in privacy occur when outlier elements, that appear in very few other players' servers, are revealed; they do not assuage such concerns. Privacy-preserving collection of statistics about computer configurations has also been considered in previous work [30, 55]. Like the work in [37], they do not give a concrete definition of security, but instead a technique for heuristically confusing attackers. Their approach also relies on chains of trust between friends, unlike our approach, in which nodes may be arbitrarily malicious. It is nearly impossible to evaluate the claims of privacy of these works, without a formal definition of security. We also believe some of the assumptions made in these works are untenable in many scenarios.

In a non-distributed context, [4, 27, 32] examine the identification of elements that appear often in a data stream, through the use of approximate counting. We generalize this task to a distributed setting, as well as enforcing important privacy properties.

Chapter 3

Preliminaries

In this thesis, we utilize several cryptographic and mathematical tools described in previous work. We briefly describe these tools in this chapter, including references to fuller descriptions, as well as the standard adversary models utilized in this thesis.

3.1 Adversary Models

In this section we describe the adversary models used in the work throughout this thesis. We provide intuition and informal definitions of these models; formal definitions can be found in [28].

3.1.1 Honest-But-Curious Adversaries

Honest-but-curious adversaries act according to their prescribed actions in the protocol. Security against such adversaries is straightforward: no player or coalition of $c < n$ honest-but-curious players (who may cheat by sharing their private information) gains information about other players' private input sets, other than what can be deduced from the result of the protocol. This is formalized by considering an ideal implementation where a trusted third party (TTP) receives the inputs of the parties and outputs the result of the defined function. We require that in the real implementation of the protocol—that is, one without a TTP—each party does not learn more information than in the ideal implementation, with overwhelming probability.

3.1.2 Malicious Adversaries

Malicious adversaries may behave arbitrarily, in contrast to honest-but-curious adversaries who follow the specified protocol. In particular, we cannot hope to prevent malicious parties from refusing to participate in the protocol, choosing arbitrary values for their private data inputs, or aborting the protocol prematurely. Instead, we focus on the standard security definition (see, e.g., [28]) which captures the correctness and the privacy issues of the protocol. Informally, the security definition is based on a comparison between the ideal model and a TTP, where a malicious party may give arbitrary input to the TTP. The security definition is also limited to the case where at least one of the parties is honest. Let Γ be the set of colluding malicious parties; for any strategy Γ can follow in the real protocol, there is a translated strategy that it could

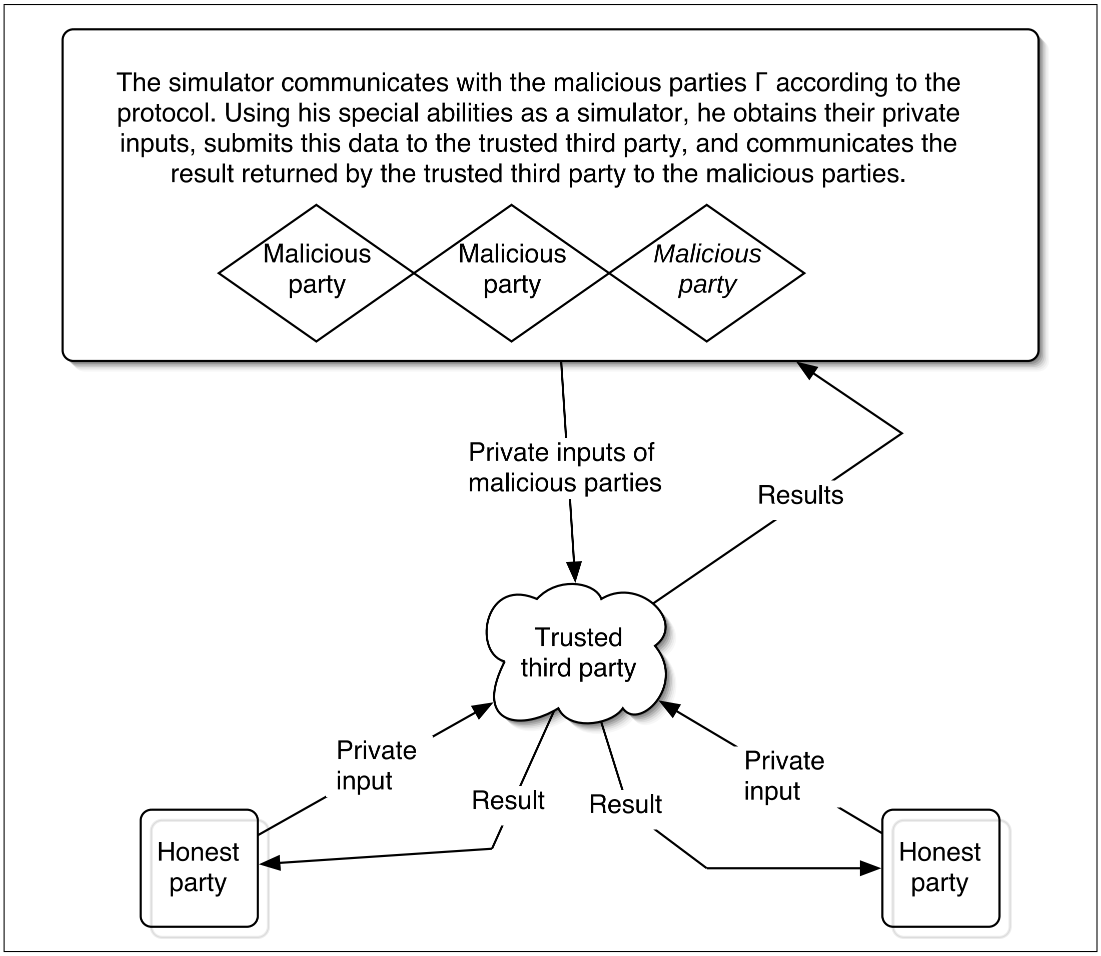

The simulator communicates with the malicious parties Γ according to the protocol. Using his special abilities as a simulator, he obtains their private inputs, submits this data to the trusted third party, and communicates the result returned by the trusted third party to the malicious parties.
Malicious party
Malicious party
Malicious party
Private inputs of malicious parties
Results
Trusted third party
Private input
Result
Result
Private input
Honest party
Honest party

Figure 3.1: Basic outline of a standard simulation proof.

follow in the ideal model, such that, to Γ, the real execution is computationally indistinguishable from execution in the ideal model.

A simulation proof is a common method of proving security under such a definition: the simulator G provides a concrete method of translating any strategy executed by Γ to a strategy in the TTP model. We illustrate such a proof in Figure 4.9.

In this thesis we consider only the class of PPT adversaries, whether they are malicious or honest-but-curious.

3.2 Multiset Operations Preliminaries

In this section, we describe several cryptographic tools that we utilize in our constructions of Chapter 4.

3.2.1 Additively Homomorphic Cryptosystem

In Chapter 4, we utilize a semantically secure [29], additively homomorphic public-key cryptosystem whose plaintext domain can be chosen to be a ring R of arbitrarily large size. Let

$E_{pk}(\cdot)$ denote the encryption function with public key pk. The cryptosystem supports the following operations, which can be performed without knowledge of the private key: (1) Given the encryptions of a and b, $E_{pk}(a)$ and $E_{pk}(b)$, we can efficiently compute the encryption of $a+b$, denoted $E_{pk}(a+b) := E_{pk}(a) +_h E_{pk}(b)$; (2) Given a constant c and the encryption of a, $E_{pk}(a)$, we can efficiently compute the encryption of ca, denoted $E_{pk}(c \cdot a) := c \times_h E_{pk}(a)$. When such operations are performed, we require that the resulting ciphertexts be re-randomized for security. In re-randomization, a ciphertext is transformed so as to form an encryption of the same plaintext, under a different random string than the one originally used. We also require that the homomorphic public-key cryptosystem support secure (n, n)-threshold decryption, i.e., the corresponding private key is shared by a group of n players, and decryption must be performed by *all* players acting together. We also require that no PPT adversary can recover the sizes of the subfields of R with greater than negligible probability.

When we utilize an additively homomorphic cryptosystem in protocols secure against malicious players, we require that: (1) the decryption protocol be secure against malicious players – typically, this is done by requiring each player to prove in zero-knowledge that he has followed the threshold decryption protocol correctly [26]; (2) efficient construction of zero-knowledge proofs of plaintext knowledge; (3) optionally, efficient construction of certain zero-knowledge proofs concerning the use of the cryptosystem's homomorphic properties, as detailed in Section 4.4.1.

Note that Paillier's cryptosystem [45] satisfies each of our requirements: it is additively homomorphic, supports ciphertext re-randomization and threshold decryption (secure in the malicious case) [21, 22], allows efficient zero-knowledge proofs for the cases that we require (these are standard constructions from [9, 14] and proof of plaintext knowledge [15]), and recovering the sizes of the subfields of the plaintext domain R is equivalent to breaking the semantic security of the cryptosystem. Key generation can be performed in a distributed fashion for these distributed Paillier schemes [21, 22].

In Chapter 4, we simply use $E_{pk}(\cdot)$ to denote the encryption function of a homomorphic cryptosystem which satisfies all the aforementioned properties.

3.2.2 Shuffle Protocol

Let each player i $(1 \leq i \leq n)$ in the Shuffle protocol have a private input multiset V_i. We define the Shuffle problem as follows: all players learn the joint multiset $V_1 \cup \cdots \cup V_n$, such that no player or coalition of $c < n$ players Γ can gain a non-negligible advantage in distinguishing, for each element $a \in V_1 \cup \cdots \cup V_n$, an honest player i $(1 \leq i \leq n,\, i \notin \Gamma)$ such that $a \in V_i$. That is, the origin of each element (contributed by an honest player) in the joint multiset $V_1 \cup \cdots \cup V_n$ is anonymous to any player or coalition of $c < n$ players. A Shuffle protocol may be secure against honest-but-curious or malicious players; we specify this security requirement in context of the protocol's use.

In several protocols in Chapter 4, we will impose an additional privacy condition on the Shuffle problem; the multisets $V_1, \ldots, V_n$ are composed of ciphertexts, which must be re-randomized so that no player may determine which ciphertexts were part of his private input multiset. The revised problem statement is as follows: all players learn the joint multiset $V_1 \cup \cdots \cup V_n$, such that no player or coalition of players can gain a non-negligible advantage in distinguishing, for each element $a \in V_1 \cup \cdots \cup V_n$, a player i $(1 \leq i \leq n)$ such that $a \in V_i$. That is, the origin of each element (contributed by any player) in the joint multiset $V_1 \cup \cdots \cup V_n$ is anonymous to any player or coalition of $c < n$ players.

Both variants of the Shuffle protocol can be easily accomplished with standard techniques [13, 17, 24, 31, 44], with communication complexity at most $O(n^2k)$.

Chapter 4

Privacy-Preserving Set and Multiset Operations

Sets and multisets are common data formats; many database operations may be represented as operations on sets and multisets. We thus examine in this chapter an important application of privacy-preserving distributed information sharing: composable privacy-preserving set and multiset operations, and secure protocols based on these operations.

We begin by describing our composable operations and their mathematical foundations in Section 4.1 before proceeding to construct protocols for several important applications. These protocols include several secure against honest-but-curious adversaries: Set-Intersection and Cardinality Set-Intersection (Section 4.2), as well as Over-Threshold Set-Union and several variants on Threshold Set-Union (Section 4.3). We then construct protocols for Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union secure against malicious players in Section 4.4. To show that our techniques extend even beyond privacy-preserving set and multiset operations, we briefly describe protocols for several additional applications in Section 4.5. In Table 4, we show the communication complexity of several of our protocols, and compare their efficiencies to that of previous work (see Section 2.1 for a more detailed discussion of previous work).

	`Our solution`	`Previous solution`	`General MPC`
Set-Intersection (HBC)	$O(cnk \lg \lvert P \rvert)$	$O(n^2 k \lg \lvert P \rvert)$ [23]	$O(n^2 k \text{ polylog}(k) \lg \lvert P \rvert)$
Set-Intersection (Malicious)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^3 k \text{ polylog}(k) \lg \lvert P \rvert)$
Cardinality Set-Intersection (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(k) \lg \lvert P \rvert)$
Over-Threshold Set-Union (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(nk) \lg \lvert P \rvert)$
Threshold Set-Union (HBC)	$O(n^2 k \lg \lvert P \rvert)$	none	$O(n^2 k \text{ polylog}(nk) \lg \lvert P \rvert)$
Subset (HBC)	$O(k \lg \lvert P \rvert)$	none	$O(k \text{ polylog}(k) \lg \lvert P \rvert)$

Figure 4.1: Total communication complexity comparison for our multiparty protocols, previous solutions, and general multiparty computation. There are $n \geq 2$ players, $c < n$ dishonestly colluding, each with an input multiset of size k. The domain of the multiset elements is P. Security parameters are not included in the communication complexity.

4.1 Techniques and Mathematical Intuition

In this section, we introduce our techniques for privacy-preserving computation of operations on sets and multisets.

Problem Setting. Let there be n players. We denote the private input set of player i as S_i, and $|S_i| = k$ $(1 \leq i \leq n)$. We denote the jth element of set i as $(S_i)_j$. We denote the domain of the elements in these sets as P, $(\forall_{i \in [n], j \in [k]} \ (S_i)_j \in P)$.

Let R denote the plaintext domain $\mathrm{Dom}(E_{pk}(\cdot))$ (in Paillier's cryptosystem, R is Z_N). We require that R be sufficiently large that an element a drawn uniformly from R has only negligible probability of *representing an element of* P, denoted $a \in P$. For example, we could require that only elements of the form $b = a \mid\mid h(a)$ could represent an element in P, where $h(\cdot)$ denotes a cryptographic hash function [40]. That is, there exists an a of proper length such that $b = a \mid\mid h(a)$. If $|h(\cdot)| = \lg\left(\frac{1}{\epsilon}\right)$, then there is only ϵ probability that $a' \leftarrow R$ represents an element in P.

In this section, we first give background on polynomial representation of multisets, as well as the mathematical properties of polynomials that we use in this chapter. We then introduce our privacy-preserving (in a TTP setting) multiset operations using polynomial representations, then show how to achieve privacy in the real setting by computing them using encrypted polynomials. Finally, we overview the applications of these techniques explored in the rest of the chapter.

4.1.1 Background: Polynomial Rings and Polynomial Representation of Sets

The polynomial ring $R[x]$ consists of all polynomials with coefficients from R. Let $f, g \in R[x]$, such that $f(x) = \sum_{i=0}^{\deg(f)} f[i]x^i$, where $f[i]$ denotes the coefficient of x^i in the polynomial f. Let $f + g$ denote the addition of f and g, $f * g$ denote the multiplication of f and g, and $f^{(d)}$ denote the dth formal derivative of f. Note that the *formal derivative* of f is $\sum_{i=0}^{\deg(f)-1}(i+1)f[i+1]x^i$.

Polynomial Representation of Sets. In this chapter, we use polynomials to represent multisets. Given a multiset $S = \{S_j\}_{1 \leq j \leq k}$, we construct a polynomial representation of S, $f \in R[x]$, as $f(x) = \prod_{1 \leq j \leq k}(x - S_j)$. On the other hand, given a polynomial $f \in R[x]$, we define the multiset S represented by the polynomial f as follows: an element $a \in S$ if and only if (1) $f(a) = 0$ and (2) a represents an element from P. Note that our polynomial representation naturally handles multisets: The element a appears in the multiset b times if $(x - a)^b \mid f \ \wedge \ (x - a)^{b+1} \not\mid f$.

Note that previous work utilized polynomials to represent sets [23] (as opposed to multisets). However, to the best of our knowledge, no operations beyond polynomial evaluation have been employed to manipulate said polynomials. As a result, previous work is limited to set intersection and cannot be composed with other set operators. In this chapter, we propose a framework to perform various set and multiset operations using polynomial representations and construct efficient privacy-preserving set operations using the mathematical properties of polynomials. By utilizing polynomial representations to represent sets and multisets, our framework allows arbitrary composition of multiset operators as outlined in our grammar.

4.1.2 Our Techniques: Privacy-Preserving Multiset Operations

In this section, we construct algorithms for computing the polynomial representation of operations on sets, including union, intersection, and element reduction. We design these algorithms to be privacy-preserving in the following sense: the polynomial representation of any operation result reveals no more information than the set representation of the result. First, we introduce our algorithms for computing the polynomial representation of set operations union, intersection, and element reduction (with a trusted third party). We then extend these techniques to encrypted polynomials, allowing secure implementation of our techniques without a trusted third party. Note that the privacy-preserving multiset operations defined in this section may be *arbitrarily composed* (see Section 4.5.1), and constitute truly general techniques.

Set Operations Using Polynomial Representations

In this section, we introduce efficient techniques for multiset operations using polynomial representations. In particular, let f, g be polynomial representations of the multisets S and T, respectively. We describe techniques to compute the polynomial representation of their union, intersection, and element reduction. We design our techniques so that the polynomial representation of any operation result reveals no more information than the multiset representation of the result. We formally state a strong privacy property for each operation in Theorems 1, 3, and 5.

Union. We define the union of multisets $S \cup T$ as the multiset where each element a that appears in S $b_S \geq 0$ times and T $b_T \geq 0$ times appears in the resulting multiset $b_S + b_T$ times. We compute the polynomial representation of $S \cup T$ as follows, where f and g are the polynomial representation of S and T respectively:

$$f * g.$$

Note that $f * g$ is a polynomial representation of $S \cup T$ because (1) all elements that appear in either set S or T are preserved: $(f(a) = 0) \wedge (g(b) = 0) \rightarrow ((f * g)(a) = 0) \wedge ((f * g)(b) = 0)$; (2) as $f(a) = 0 \Leftrightarrow (x - a) \mid f$, duplicate elements from each multiset are preserved: $(f(a) = 0) \wedge (g(a) = 0) \rightarrow (x - a)^2 \mid (f * g)$. In addition, we prove that, given $f * g$, one cannot learn more information about S and T than what can be deduced from $S \cup T$, as formally stated in the following theorem:

Theorem 1. *Let* TTP1 *be a trusted third party which receives the private input multiset S_i from player i for $1 \leq i \leq n$, and then returns to every player the union multiset $S_1 \cup \cdots \cup S_n$ directly. Let* TTP2 *be another trusted third party, which receives the private input multiset S_i from player i for $1 \leq i \leq n$, and then: (1) calculates the polynomial representation f_i for each S_i; (2) computes and returns to every player $\prod_{i=1}^{n} f_i$.*

There exists a PPT translation algorithm such that, to each player, the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof. Theorem 1 is trivially true. (This theorem is included for completeness.) □

Intersection. We define the intersection of multisets $S \cap T$ as the multiset where each element a that appears in S $b_S > 0$ times and T $b_T > 0$ times appears in the resulting multiset

$\min\{b_S, b_T\}$ times. Let S and T be two multisets of equal size, and f and g be their polynomial representations (also of equal size) respectively. We compute the polynomial representation of $S \cap T$ as:

$$f * r + g * s$$

where $r, s \leftarrow R^{\deg(f)}[x]$, where $R^b[x]$ is the set of all polynomials of degree $0, \ldots, b$ with coefficients chosen independently and uniformly from R: $r = \sum_{i=0}^{\deg(f)} r[i]x^i$ and $s = \sum_{i=0}^{\deg(f)} s[i]x^i$, where $\forall_{0 \le i \le \deg(f)}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \deg(f)}\ s[i] \leftarrow R$.

We show below that $f * r + g * s$ is a polynomial representation of $S \cap T$. In addition, we prove that, given $f * r + g * s$, one cannot learn more information about S and T than what can be deduced from $S \cap T$, as formally stated in Theorem 3.

First, we must prove the following lemma, based on our definition of gcd as the output of Euclid's gcd algorithm (see Lemma 19 in Section 4.6):

Lemma 2. *Let f, g be polynomials in $R[x]$ where R is a ring such that no PPT adversary can find the size of its subfields with non-negligible probability,* $\deg(f) = \deg(g) = \alpha$, $\beta \ge \alpha$, $\gcd(f, g) = 1$, *and* $f[\deg(f)] \in R^* \wedge g[\deg(g)] \in R^*$. *Let $r = \sum_{i=0}^{\beta} r[i]x^i$ and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \le i \le \beta}\ r[i] \leftarrow R$, $\forall_{0 \le i \le \beta}\ s[i] \leftarrow R$ (independently).*

Let $u = f * r + g * s = \sum_{i=0}^{\alpha+\beta} u[i]x^i$. Then $\forall_{0 \le i \le \alpha+\beta}\ u[i]$ are distributed uniformly and independently over R.

We prove Lemma 2 in Section 4.6.

By this lemma, $f * r + g * s = \gcd(f, g) * u$, where u is distributed uniformly in $R^{\gamma}[x]$ for $\gamma = 2\deg(f) - |S \cap T|$. Note that a is a root of $\gcd(f, g)$ and $(x - a)^{\ell_a} \mid \gcd(f, g)$ if and only if a appears ℓ_a times in $S \cap T$. Moreover, because u is distributed uniformly in $R^{\gamma}[x]$, with overwhelming probability the roots of u do not represent any element from P (as explained in the beginning of Section 4.1). Thus, the computed polynomial $f * r + g * s$ is a polynomial representation of $S \cap T$. Note that this technique for computing the intersection of two multisets can be extended to simultaneously compute the intersection of an arbitrary number of multisets in a similar manner. Also, given $f * r + g * s$, one cannot learn more information about S and T than what can be deduced from $S \cap T$, as formally stated in the following theorem:

Theorem 3. *Let* TTP1 *be a trusted third party which receives the private input multiset S_i of size k from player i for $1 \le i \le n$, and then returns to every player the intersection multiset $S_1 \cap \cdots \cap S_n$ directly. Let* TTP2 *be another trusted third party, which receives the private input multiset S_i from player i for $1 \le i \le n$, and then: (1) calculates the polynomial representation f_i for each S_i; (2) chooses $r_i \leftarrow R^k[x]$; (3) computes and returns to each player $\sum_{i=1}^{n} f_i * r_i$.*

There exists a PPT translation algorithm such that, to each player, the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof sketch. Let the output of TTP1 be denoted T. The translation algorithm operates as follows: (1) calculates the polynomial representation g of T; (2) chooses the random polynomial $u \leftarrow R^{2k-|T|}[x]$; (3) computes and returns $g * u$. □

Element Reduction. We define the operation of element reduction (by d) of a multiset S (denoted $\mathrm{Rd}_d(S)$) as follows: for each element a that appears b times in S, it appears

$\max\{b-d, 0\}$ times in the resulting multiset. We compute the polynomial representation of $\mathrm{Rd}_d(S)$ as:

$$\sum_{j=0}^{d} f^{(j)} * F_j * r_j$$

where $r_j \leftarrow R^{\deg(f)}[x]$ $(0 \leq j \leq d)$ and each F_j is any polynomial of degree j, such that $\forall_{a \in P}\ F(a) \neq 0$ $(0 \leq j \leq d)$ and $\gcd(F_0, \ldots, F_d) = 1$. Note that random polynomials of degree $0, \ldots, d$ in $R[x]$ have these properties with overwhelming probability.

To show that formal derivative operation allows element reduction, we require the following lemma:

Lemma 4. *Let $F_j \in R[x]$ $(0 \leq j \leq d)$ each of degree j such that $\gcd(F_0, \ldots, F_d) = 1$. For all elements $a \in R$ such that $\forall_{0 \leq j \leq d}\ (x-a) \nmid F_j$, $q \in R[X]$ such that $(x-a) \nmid q$, and $r_j \leftarrow R^{m+\deg(q)}[x]$ $(0 \leq j \leq d)$, and:*

- *if $m > d$, $f = (x-a)^m * q \rightarrow (x-a)^{m-d} \mid \sum_{j=0}^{d} f^{(j)} * F_j * r_j \quad \wedge \quad (x-a)^{m-d+1} \nmid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$*
- *if $m \leq d$, $f = (x-a)^m * q \rightarrow (x-a) \nmid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$*

with overwhelming probability.

We prove this lemma in Section 4.6. By Lemma 2, $\sum_{j=0}^{d} f^{(j)} * F_j * r_j = \gcd(f^{(d)}, f^{(d-1)}, \ldots, f) * u$, where u is distributed uniformly in $R^{\gamma}[x]$ for $\gamma = 2k - |\mathrm{Rd}_d(S)|$. Thus, with overwhelming probability, any root of u does not represent any element from P. Therefore, $\sum_{j=0}^{d} f^{(j)} * F_j * r_j$ is a polynomial representation of $\mathrm{Rd}_d(S)$, and moreover, given $\sum_{j=0}^{d} f^{(j)} * F_j * r_j$, one cannot learn more information about S than what can be deduced from $\mathrm{Rd}_d(S)$, as formally stated in the following theorem:

Theorem 5. *Let F_j $(0 \leq j \leq d)$ be publicly known polynomials of degree j such that $\forall_{a \in P}\ F_j(a) \neq 0$ and $\gcd(F_0, \ldots, F_d) = 1$. Let* TTP1 *be a trusted third party which receives a private input multiset S of size k, and then returns the reduction multiset $Rd_d(S)$ directly. Let* TTP2 *be another trusted third party, which receives a private input multiset S, and then: (1) calculates the polynomial representation f of S; (2) chooses $r_0, \ldots, r_d \leftarrow R^k[x]$; (3) computes and returns $\sum_{j=0}^{d} f^{(j)} * F_j * r_j$.*

There exists a PPT translation algorithm such that the results of the following two scenarios are distributed identically: (1) applying translation to the output of TTP1*; (2) returning the output of* TTP2 *directly.*

Proof sketch. Let the output of TTP1 be denoted T. The translation algorithm operates as follows: (1) calculates the polynomial representation g of T; (2) chooses the random polynomial $u \leftarrow R^{2k-|T|}[x]$; (3) computes and returns $g * u$. □

Operations with Encrypted Polynomials

In the previous section, we prove the security of our polynomial-based multiset operators when the polynomial representation of the result is computed by a trusted third party (TTP2). By using additively homomorphic encryption, we allow these results to be implemented as protocols in the real world without a trusted third party (i.e., the polynomial representation of the set operations is computed by the parties collectively without a trusted third party). In the algorithms given above, there are three basic polynomial operations that are used: addition, multiplication, and the formal derivative. We give algorithms in this section for computation of these operations with encrypted polynomials.

For $f \in R[x]$, we represent the *encryption of polynomial* f, $E_{pk}(f)$, as the ordered list of the encryptions of its coefficients under the additively homomorphic cryptosystem: $E_{pk}(f[0]), \ldots, E_{pk}(f[\deg(f)])$. Let f_1, f_2, and g be polynomials in $R[x]$ such that $f_1(x) = \sum_{i=0}^{\deg(f_1)} f_1[i]x^i$, $f_2(x) = \sum_{i=0}^{\deg(f_2)} f_2[i]x^i$, and $g(x) = \sum_{i=0}^{\deg(g)} g[i]x^i$. Let $a, b \in R$. Using the homomorphic properties of the homomorphic cryptosystem, we can efficiently perform the following operations on encrypted polynomials without knowledge of the private key:

- Sum of encrypted polynomials: given the encryptions of the polynomial f_1 and f_2, we can efficiently compute the encryption of the polynomial $g := f_1 + f_2$, by calculating $E_{pk}(g[i]) := E_{pk}(f_1[i]) +_h E_{pk}(f_2[i])$ $(0 \le i \le \max\{\deg(f_1), \deg(f_2)\})$

- Product of an unencrypted polynomial and an encrypted polynomial: given a polynomial f_2 and the encryption of polynomial f_1, we can efficiently compute the encryption of polynomial $g := f_1 * f_2$, (also denoted $f_2 *_h E_{pk}(f_1)$) by calculating the encryption of each coefficient $E_{pk}(g[i]) := (f_2[0] \times_h E_{pk}(f_1[i])) +_h (f_2[1] \times_h E_{pk}(f_1[i-1])) +_h \ldots +_h (f_2[i] \times_h E_{pk}(f_1[0]))$ $(0 \le i \le \deg(f_1) + \deg(f_2))$.

- Derivative of an encrypted polynomial: given the encryption of polynomial f_1, we can efficiently compute the encryption of polynomial $g := \frac{d}{dx} f_1$, by calculating the encryption of each coefficient $E_{pk}(g[i]) := (i + 1) \times_h E_{pk}(f_1[i + 1])$ $(0 \le i \le \deg(f_1) - 1)$.

- Evaluation of an encrypted polynomial at an unencrypted point: given the encryption of polynomial f_1, we can efficiently compute the encryption of $a := f_1(b)$, by calculating $E_{pk}(a) := (b^0 \times_h E_{pk}(f_1[0])) +_h (b^1 \times_h E_{pk}(f_1[1])) +_h \ldots +_h (b^{\deg(f_1)} \times_h E_{pk}(f_1[\deg(f_1)]))$.

Utilizing the above operations on encrypted polynomials, we can securely compute results according to the multiset operations described in Section 4.1.2 without the trusted third party (TTP2). We demonstrate this property with concrete examples detailed in the remainder of this chapter.

4.1.3 Overview of Applications

The techniques we introduce for privacy-preserving computations of multiset operations have many applications. We give several concrete examples that utilize our techniques for specific privacy-preserving functions on multisets in the following sections.

First, we design efficient protocols for the Set-Intersection and Cardinality Set-Intersection problems, secure against honest-but-curious adversaries (Section 4.2). We then provide an efficient protocol for the Over-Threshold Set-Union problem, as well as three variants of the Threshold Set-Union problem, secure against honest-but-curious adversaries, in Section 4.3. We introduce tools and protocols, secure against malicious players, for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems in Section 4.4. We propose an efficient protocol for the Subset problem in Section 4.5.2.

Protocol: Set-Intersection-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key of a homomorpic cryptosystem.
Output: Each player determines $S_1 \cap \cdots \cap S_n$.

1. Each player $i = 1, \ldots, n$
   (a) calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends the encryption of the polynomial f_i to players $i+1, \ldots, i+c$
   (c) chooses $c+1$ polynomials $r_{i,0}, \ldots, r_{i,c} \leftarrow R^k[x]$
   (d) calculates the encryption of the polynomial $\phi_i = f_{i-c} * r_{i,i-c} + \cdots + f_{i-1} * r_{i,i-1} + f_i * r_{i,0}$, utilizing the algorithms given in Sec. 4.1.2.
2. Player 1 sends the encryption of the polynomial $\lambda_1 = \phi_1$, to player 2
3. Each player $i = 2, \ldots, n$ in turn
   (a) receives the encryption of the polynomial λ_{i-1} from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} + \phi_i$ by utilizing the algorithms given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ to all other players.
5. All players perform a group decryption to obtain the polynomial p.

Each player $i = 1, \ldots, n$ determines the intersection multiset as follows: for each $a \in S_i$, he calculates b such that $(x-a)^b | p \ \wedge \ (x-a)^{b+1} \not| p$. The element a appears b times in the intersection multiset.

Figure 4.2: Set-Intersection protocol secure against honest-but-curious adversaries.

More generally, our techniques allow private computation of functions based on composition of the union, intersection, and element reduction operators. We discuss techniques for this general private computation on multisets in Section 4.5.1.

Our techniques are widely applicable, even outside the realm of computation of functions over multisets. As an example, we show how to apply our techniques to private evaluation of boolean formulae in CNF form in Section 4.5.3.

4.2 Application I: Private Set-Intersection and Cardinality Set-Intersection

In this section, we design protocols for Set-Intersection and Cardinality Set-Intersection secure against a coalition of honest-but-curious adversaries.

4.2.1 Set-Intersection

Problem Definition. Let there be n parties; each has a private input set S_i $(1 \leq i \leq n)$ of size k. We define the *Set-Intersection* problem as follows: all players learn the intersection of all private input multisets without gaining any other information; that is, each player learns $S_1 \cap S_2 \cap \cdots \cap S_n$.

Our protocol secure against honest-but-curious adversaries is given in Fig. 4.2. In this protocol, each player i $(1 \leq i \leq n)$ first calculates a polynomial representation $f_i \in R[x]$ of his input multiset S_i. He then encrypts this polynomial f_i, and sends it to c other players $i+1, \ldots, i+c$. For each encrypted polynomial $E_{pk}(f_i)$, each player $i+j$ $(0 \leq j \leq c)$ chooses a random polynomial $r_{i+j,j} \in R^k[x]$. Note that at most c players may collude, thus $\sum_{j=0}^{c} r_{i+j,j}$ is both uniformly distributed and known to no player. They then compute the encrypted polynomial $\left(\sum_{j=0}^{c} r_{i+j,j}\right) *_h E_{pk}(f_i)$. From these encrypted polynomials, the players compute the encryption of $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$. All players engage in group decryption to obtain the polynomial p. Thus, by Theorem 3, the players have privately computed p, a polynomial representing the intersection of their private input multisets. Finally, to reconstruct the multiset represented by polynomial p, the player i, for each $a \in S_i$, calculates b such that $(x-a)^b | p \ \wedge \ (x-a)^{b+1} \not| p$. The element a appears b times in the intersection multiset.

Security Analysis. We show that our protocol is correct, as each player learns the appropriate answer set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model. A formal statement of these properties is as follows:

Theorem 6. *In the Set-Intersection protocol of Fig. 4.2, every player learns the intersection of all players' private inputs, $S_1 \cap S_2 \cap \cdots \cap S_n$, with overwhelming probability.*

Proof. Each player learns the decrypted polynomial $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$. If $\forall_{i\in[n]}\ f_i(a) = 0$, then $p(a) = 0$. As no elements that are not in every players' private input can be in the set-intersection of all private inputs, all elements in the set-intersection can be recovered by each player. Each element in his private input that a root of p is a member of the intersection set.

We now show that, with high probability, erroneous elements are not inserted into the answer set. Note that, by the reasoning of Lemma 19, all coefficients of f_i $(1 \le i \le n)$ are in the set $R^* \cup \{0\}$. Thus, by Lemma 2, the decrypted polynomial is of the form $\left(\prod_{a\in I}(x-a)\right) * s$, where s is uniformly distributed over $R^{2k-|I|}[x]$. This random polynomial s is of polynomial size, and thus has a polynomial number of roots. Each of these roots is a representation of an element from P with only negligible probability. Thus, the probability that an erroneous element is included in the answer set is also negligible, and all players learn exactly the intersection set. □

Theorem 7. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, with overwhelming probability, in the Set-Intersection protocol of Fig. 4.2, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

Proof. We assume that the homomorphic cryptosystem (E, D) used in the protocol is in fact secure as we required. Thus, as the inputs of the other players are all encrypted until the decryption is performed, nothing can be learned by any player before that point. Each player j then learns only the summed polynomial $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$.

Note that to every coalition of c players, for every i, $\sum_{j=0}^{c} r_{i+j,j}$ is completely random, as at least one player in the $c+1$ players who chose that random polynomial is not a member of the coalition, and so $\sum_{j=0}^{c} r_{i+j,j}$ is uniformly distributed and unknown.

Note that, by the reasoning of Lemma 19, all coefficients of f_i $(1 \le i \le n)$ are in the set $R^* \cup \{0\}$. Thus, by Lemma 2, $p = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right) = \left(\prod_{a\in I}(x-a)\right) * s$, where I is the intersection set and s is uniformly distributed over the polynomials of appropriate degree.

Protocol: Cardinality-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key of a homomorpic cryptosystem.
Output: Each player determines $|S_1 \cap \cdots \cap S_n|$.

1. Each player $i = 1, \ldots, n$
   (a) calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends the encryption of the polynomial f_i to players $i + 1, \ldots, i + c$
   (c) chooses $c + 1$ random polynomials $r_{i,0}, \ldots, r_{i,c} \leftarrow R^k[x]$
   (d) calculates the encryption of the polynomial $\phi_i = f_{i-c} * r_{i,i-c} + \cdots + f_{i-1} * r_{i,i-1} + f_i * r_{i,0}$, utilizing the algorithms given in Sec. 4.1.2.
2. Player 1 sends the encrypted polynomial $\lambda_1 = \phi_1$, to player 2
3. Each player $i = 2, \ldots, n$ in turn
   (a) receives the encryption of the polynomial λ_{i-1} from player $i - 1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} + \phi_i$ by utilizing the algorithms given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial λ_i to player $i + 1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \sum_{i=1}^{n} f_i * \left(\sum_{j=0}^{c} r_{i+j,j}\right)$ to all other players.
5. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial p at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.1.2.
   (b) for each $j = 1, \ldots, k$ chooses a random number $r_{ij} \leftarrow R$ and calculates an encrypted element $(V_i)_j = r_{ij} \times_h E_{pk}(c_{ij})$
6. All players perform the Shuffle protocol on their private input sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized.
7. All players $1, \ldots n$ decrypt each element of the shuffled set V.

If nb of the decrypted elements from V are 0, then the size of the set intersection is b.

Figure 4.3: Cardinality set-intersection protocol secure against honest-but-curious adversaries.

Thus no information about the private inputs of the honest players can be recovered from p, other than that given by revealing the intersection set. □

4.2.2 Cardinality Set-Intersection

Problem Definition. We define the Cardinality Set-Intersection problem on sets as follows: each player learns the number of unique elements in $S_1 \cap \cdots \cap S_n$, without learning any other information. A variant of this problem is the Cardinality Set-Intersection problem on multisets, which we define as follows: all players learn $|S_1 \cap \cdots \cap S_n|$, as computed on multisets.

Our protocol for Cardinality Set-Intersection, given in Figure 4.3, proceeds as our protocol for Set-Intersection, until the point where all players learn the encryption of p, the polynomial representation of $S_1 \cap \cdots \cap S_n$. Each player $i = 1, \ldots, n$ then evaluates this encrypted polynomial at each unique element $a \in S_i$, obtaining β_a, an encryption of $p(a)$. He then blinds each encrypted evaluation $p(a)$ by calculating $\beta'_a = b_a \times_h \beta_a$. All players then distribute and shuffle the ciphertexts β'_a constructed by each player, such that all players receive all ciphertexts, without learning their source. The Shuffle protocol can be constructed from standard techniques [13, 17, 24, 31, 44], with communication complexity at most $O(n^2k)$. The players then decrypt these ciphertexts, finding that nb of the decryptions are 0, implying that there are b unique elements in $S_1 \cap \cdots \cap S_n$. FNP utilize a variation of this technique [23], but it is not obvious how to construct a multiparty Cardinality Set-Intersection protocol from their techniques.

Variants. Our protocol can be simply extended to privately compute the Cardinality Set-Intersection problem on multisets, by utilizing an encoding as follows: any element a that appears b times in a multiset is encoded as the set: $\{a \mid\mid 1, \ldots, a \mid\mid b\}$, with element included only once. Note that this is a set of equivalent size as the original multiset representation, so this variant preserves the efficiency of our protocol.

Security Analysis. We show that our protocol is correct, as each player learns the size of the answer set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model. A formal statement of these properties is as follows:

Theorem 8. *In the Cardinality Set-Intersection protocol of Fig. 4.3, every player learns the size of the intersection of all players' private inputs, $|S_1 \cap S_2 \cap \cdots \cap S_n|$, with overwhelming probability.*

Proof. Note that, following the proof of Theorem 6, p is a polynomial representation of the intersection multiset, with overwhelming probability. Each player evaluates p (encrypted) at each of their inputs, then blinds it by homomorphically multiplying a random element by the encrypted evaluation. Thus each resulting encrypted element $(V_i)_j$ $(1 \le i \le n,\ 1 \le j \le k)$ is either 0, representing some element of a private input set in the intersection set, or uniformly distributed, representing some element not in the intersection set. An element is a member of $S_1 \cap \cdots \cap S_n$ if and only if each player holds it as part of their private input set, for each element of $S_1 \cap \cdots \cap S_n$, there are n encrypted evaluations that are 0. Thus, when the encrypted evaluations $(V_i)_j$ $(1 \le i \le n,\ 1 \le j \le k)$ are shuffled and decrypted, there are exactly $n|S_1 \cap \cdots \cap S_n|$ 0s, and thus all players learn the size of the intersection set. □

Theorem 9. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Cardinality Set-Intersection protocol of Fig. 4.3, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

Proof. We assume that the cryptosystem $E_{pk}(\cdot)$ and Shuffle protocol are secure, so we may note that no player or coalition of players learns any information from the protocol except the decryption of the randomly-ordered set $\{(V_i)_j\}_{i \in [n], j \in [k]}$. As each element of that set is either 0 or a uniformly distributed element, it conveys no information other than the statement 'some player had an element in their private input set that was/was not in the intersection set'. As this information precisely constitutes the result of the Cardinality Set-Intersection problem, no additional information is revealed. □

4.2.3 Malicious Case

We can extend our protocols in Figures 4.2 and 4.3, secure against honest-but-curious players, to protocols secure against malicious adversaries by adding zero-knowledge proofs or using cut-and-choose to ensure security. We give details of our protocols secure against malicious adversaries in Section 4.4.2.

Protocol: Over-Threshold Set-Union-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key for a homomorphic cryptosystem. The threshold number of repetitions at which an element appears in the output is t. $F_0, \ldots, F_{t-1}$ are fixed polynomials of degree $0, \ldots, t-1$ which have no common factors or roots representing elements of P.
Output: Each player determines $\text{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \ldots, n$
    (a) receives the encryption of the polynomial λ_{i-1} from player $i - 1$
    (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} * f_i$ by utilizing the algorithm given in Sec. 4.1.2.
    (c) sends the encryption of the polynomial λ_i to player $i + 1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \prod_{i=1}^{n} f_i$ to players $2, \ldots, c+1$
5. Each player $i = 1, \ldots, c+1$
    (a) calculates the encryption of the $1, .., t-1$st derivatives of p, denoted $p^{(1)}, \ldots, p^{(t-1)}$, by repeating the algorithm given in Sec. 4.1.2.
    (b) chooses random polynomials $r_{i,0}, \ldots, r_{i,t-1} \leftarrow R^{nk}[x]$
    (c) calculates the encryption of the polynomial $\sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * r_{i,\ell}$ and sends it to all other players.
6. All players perform a group decryption to obtain the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_j * \left(\sum_{i=0}^{c+1} r_{i,\ell}\right)$.
7. Each player $i = 1, \ldots, n$, for each $j = 1, \ldots, k$
    (a) chooses a random element $b_{i,j} \leftarrow R$
    (b) calculates $u_{i,j} = b_{i,j} \times \Phi((S_i)_j) + (S_i)_j$
8. All players $i = 1, \ldots n$ perform the Shuffle protocol on the elements $u_{i,j}$ $(1 \leq j \leq k)$, such that each player obtains a joint set V.

Each element $a \in P$ that appears b times in V is an element in the threshold set that appears b times in the players' private inputs.

Figure 4.4: Over-Threshold Set-Union protocol secure against honest-but-curious adversaries.

4.3 Application II: Private Over-Threshold Set-Union and Threshold Set-Union

In this section, we design protocols for the Over-Threshold Set-Union problem and several variations of the Threshold Set-Union problem, secure against a coalition of honest-but-curious adversaries.

4.3.1 Over-Threshold Set-Union Protocol

Problem Definition. Let there be n players; each has a private input set S_i $(1 \leq i \leq n)$ of size k. We define the *Over-Threshold Set-Union* problem as follows: all players learn which elements appear in the union of the players' private input multisets at least a threshold number t times, and the number of times these elements appeared in the union of players' private inputs, without gaining any other information. For example, assume that a appears in the combined private input of the players 15 times. If $t = 10$, then all players learn a has appeared 15 times. However, if $t = 16$, then no player learns a appears in any player's private input. This problem can be represented as $\text{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

We describe our protocol secure against honest-but-curious players for the Over-Threshold Set-Union problem in Fig. 4.4. In this protocol, each player i $(1 \leq i \leq n)$ first calculates f_i, the polynomial representation of its input multiset S_i. All players then compute the encryption of polynomial $p = \prod_{i=1}^{n} f_i$, the polynomial representation of $S_1 \cup \cdots \cup S_n$. Players $i = 1, \ldots, c+1$ then each choose random polynomials $r_{i,0}, \ldots, r_{i,t-1}$, and calculate the encryption of the polynomial $\sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * r_{i,\ell}$ as shown in Fig. 4.4. All players then calculate the encryption of the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=0}^{c+1} r_{i,\ell}\right)$ and perform a group decryption to obtain Φ. As at most c players may dishonestly collude, the polynomials $\sum_{i=1}^{c+1} r_{i,\ell}$ $(1 \leq \ell \leq d)$ are uniformly distributed and known to no player. By Theorem 5, Φ is a polynomial representation of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

Each player $i = 1, \ldots, n$ then chooses $b_{i,j} \leftarrow R$ and computes $u_{i,j} = b_{i,j} \times \Phi((S_i)_j) + (S_i)_j$ $(1 \leq j \leq k)$. Each element $u_{i,j}$ equals $(S_i)_j$ if $(S_i)_j \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, and is otherwise uniformly distributed over R. The players then shuffle these elements $u_{i,j}$, such that each player learns all of the elements, but does not learn which player's set they came from. The shuffle can be easily accomplished with standard techniques [13, 17, 24, 31, 44], with communication complexity at most $O(n^2k)$. The multiset formed by those shuffled elements that represent elements of P is $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

Security Analysis. We show that our protocol is correct, as each player learns the appropriate answer set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model with a trusted third party. A formal statement of these properties is as follows:

Theorem 10. *In the Over-Threshold Set-Union protocol of Fig. 4.4, every honest-but-curious player learns each element a which appears at least t times in the union of the n players' private inputs, as well as the number of times it so appears, with overwhelming probability.*

Proof. All players calculate and decrypt $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=1}^{c+1} r_{i,\ell}\right)$. As $\sum_{i=1}^{c+1} r_{i,\ell}$ $(0 \leq \ell \leq t-1)$ are distributed uniformly over all polynomials of approximate size nk and, by the reasoning of Lemma 19, all coefficients of $p^{(\ell)} * F_\ell$ $(0 \leq \ell \leq t-1)$ are in the set $R^* \cup \{0\}$, Lemma 2 tells us that $\Phi = \gcd\left(p^{(t-1)}, p^{(t-2)}, \ldots, p\right) * u$, where u is a random polynomial of the appropriate size. As u has only a polynomial number of roots, each of which has a negligable probability of representing a member of P, u is a polynomial representation of the empty set with overwhelming probability.

By Theorem 4, $\gcd\left(p^{(t-1)}, p^{(t-2)}, \ldots, p\right)$ has roots which are exactly those that appear at least t times in the players' private inputs (the threshold set). The players calculate elements $u_{i,j}$, which are uniformly distributed if $(S_i)_j$ is not a member of the threshold set, and $(S_i)_j$ if it does appear in the threshold set. These elements are shuffled and distributed to all players. Each reveals an element of the private input, if that element is in the threshold set, and nothing otherwise. Thus each element in the threshold intersection set is revealed as many times as it appeared in the private inputs. □

Theorem 11. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, with overwhelming probability, in the Over-Threshold Set-Union protocol of Fig. 4.4, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

Proof. We assume that the cryptosystem employed is semantically secure, and so players learn only the formula $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=1}^{c+1} r_{i,\ell}\right)$. Note that $\sum_{i=1}^{c+1} r_{i,\ell}$ $(0 \leq$

$\ell \leq t-1$) are uniformly distributed and unknown to all players, as the maximum coalition size is smaller than $c+1$. Note that by the reasoning of Lemma 19, all coefficients of $p^{(\ell)} * F_\ell$ ($0 \leq \ell \leq t-1$) are in the set $R^* \cup \{0\}$. Thus, by Theorem 2, $\Phi = \gcd\left(p^{(t-1)} * F_{t-1}, p^{(t-2)} * F_{t-2}, \ldots, p * F_0\right) * s$, for some uniformly distributed polynomial s. As s is uniformly distributed for any player inputs, no player or coalition can learn more than $\gcd\left(p^{(t-1)}, p^{(t-2)}, \ldots, p\right)$. $F_0, \ldots, F_{t-1}$ are chosen such that $\gcd(p, F_0, \ldots, F_{t-1}) = 1$ with overwhelming probability, and so $\gcd\left(p^{(t-1)} * F_{t-1}, p^{(t-2)} * F_{t-2}, \ldots, p * F_0\right) = \gcd\left(p, p^{(t-1)} * F\right) = \gcd\left(p^{(t-1)}, p^{(t-2)}, \ldots, p\right)$ with overwhelming probability. As was observed in Theorem 10, this information exactly represents the threshold set, and can thus be derived from the answer that would be returned by a trusted third party. Thus no player or coalition of at most c players can learn more than in the ideal model.

Neither do the shuffled elements reveal additional information. As we assume the shuffling protocol is secure, the origin of any element is not revealed. The elements revealed are exactly those in the threshold set, each included as many times as it was included in the private inputs, and thus also do not reveal information to any adversary. □

4.3.2 Threshold Set-Union

Problem Definition. We define the Threshold Set-Union problem as follows: all players learn which elements appear in the combined private input of the players at least a threshold number t times. For example, assume that a appears in the combined private input of the players 15 times. If $t = 10$, then all players learn a. However, if $t = 16$, then no player learns a. This problem differs from the Over-Threshold Set-Union problem in that each player learns the elements of $\mathrm{Rd}_{t-1}(S_1 \cap \cdots \cap S_n)$, without learning how often each element appears.

We offer protocols for several variants on Threshold Set-Union: threshold contribution, perfect, and semi-perfect. Threshold contribution allows for thresholds $t \geq 1$, and each player learns only those elements which appear both in his private input and the threshold set: player i ($1 \leq i \leq n$) learns the elements of $S_i \cap \mathrm{Rd}_{t-1}(S_1 \cap \cdots \cap S_n)$. Perfect threshold set-intersection allows for thresholds $t \geq 1$, and conforms exactly to the definition of threshold set-intersection. The semi-perfect variant requires for security that $t \geq 2$, and that the cheating coalition does not include any single element more than $t-1$ times in their private inputs. Note that the information illicitly gained by the coalition when they include more than $t-1$ copies of an element a is restricted to a possibility of learning that there exists some other player whose private input contains a. We do not consider the difference in security between the semi-perfect and perfect variants to be significant.

The protocols for the Threshold Set-Union problem, given in Figs. 4.5, 4.6, and 4.7, are identical to the protocol for Over-Threshold Set-Union (given in Fig. 4.4) from step 1-5. We explain the differences between the protocols for each variant: threshold contribution, semi-perfect, and perfect. Each player constructs encryptions of the elements $\Phi((S_i)_j)$ from his private input set in step 6, and continues as described below.

Threshold Contribution Threshold Set-Union. This protocol is given in Fig. 4.6. The players cooperatively decrypt the encrypted elements $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$. This decryption must take place in such a way that only player i learns the element $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$. Typically, parties produce decryption shares and reconstruct the element from them; player i simply retains his decryption share, so that only he learns the decryption. Thus each player learns which of his elements appear in the threshold set, since if $(S_i)_j$ appears in the threshold

```
Protocol: THRESHOLD-SEMIPERFECT-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set $S_i$, such that $|S_i| = k$. The players share the secret key $sk$, to which $pk$ is the corresponding public key for a homomorphic cryptosystem. The threshold number of repetitions at which an element appears in the output is $t$. $F_0, \ldots, F_{t-1}$ are fixed polynomials of degree $0, \ldots, t-1$ which have no common factors or roots representing elements of $P$.
Output: Each player learns the elements of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \ldots, n$
   (a) receives the encryption of the polynomial $\lambda_{i-1}$ from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} * f_i$ by utilizing the algorithm given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial $\lambda_i$ to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \prod_{i=1}^{n} f_i$ to players $2, \ldots, c+1$
5. Each player $i = 1, \ldots, c+1$
   (a) calculates the encryption of the $1, .., t-1$st derivatives of $p$, denoted $p^{(1)}, \ldots, p^{(t-1)}$, by repeating the algorithm given in Sec. 4.1.2.
   (b) chooses random polynomials $r_{i,0}, \ldots, r_{i,t-1} \leftarrow R^{nk}[x]$
   (c) calculates the encryption of the polynomial $\sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * r_{i,\ell}$ and sends it to all other players.
6. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=1}^{c+1} r_{i,\ell}\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.1.2
   (b) for each $j = 1, \ldots, k$ calculates an encrypted tag $T_{ij} = Enc_i(h((S_i)_j) \mid\mid (S_i)_j)$
   (c) for each $j = 1, \ldots, k$ chooses a random number $r_{ij} \leftarrow R$ and calculates an encrypted element $U_{ij} = (r_{ij} \times_h E_{pk}(c_{ij})) +_h E_{pk}((S_i)_j)$
   (d) constructs the set $V_i = \{(T_{ij} \mid\mid U_{ij}) \mid 1 \leq j \leq k\}$
7. By using the Shuffle protocol, players perform shuffling on their private input sets $V_i$.
8. For each shuffled element $T \mid\mid U$ in sorted order, each player $i = 1, \ldots, n$
   (a) if $D_i(T) = h(a) \mid\mid a$ for some $a$
       i. if $a$ has previously been revealed to be in the threshold set, then calculate an incorrect decryption share of $U$, and send it to all other players
   (b) else calculate a decryption share of $U$, and send it to all other players
   (c) reconstruct the decryption of $U$. If the element $a \in P$, then $a$ is in the threshold result set
```

Figure 4.5: Threshold Set-Union protocol secure against honest-but-curious adversaries (semi-perfect variant).

set, $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j}) = 0$. No player learns more information because if an element $(S_i)_j$ is not in the threshold set, $\Phi((S_i)_j) * (\sum_{\ell=1}^{n} b_{\ell,i,j})$ is uniformly distributed.

Semi-Perfect Threshold Set-Union. This protocol is given in Fig. 4.5. The encrypted element $(U_i)_j$ calculated from the encrypted evaluation of $\Phi((S_i)_j)$ is either: (1) an encryption of the private input element $(S_i)_j$ (if $(S_i)_j$ is in the intersection set) or (2) an encryption of a random element (otherwise). However, the player also constructs a corresponding encrypted tag for each $(U_i)_j$, T_{ij}. We require that the cryptosystem used to construct these tags be key-private, so that the origin of ciphertext pairs T, U cannot be ascertained by the key used to construct the tags.

The players then correctly obtain a decryption of each element in the threshold set exactly once. Any other time a ciphertext U for an element in the threshold set is decrypted, a player

Protocol: Threshold-Contribution-HBC

Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key for a homomorphic cryptosystem. The threshold number of repetitions at which an element appears in the output is t. F is a fixed polynomial of degree $t-1$ which has no roots representing elements of P. The threshold number of repetitions at which an element appears in the output is $t \geq 2$. $F_0, \dots, F_{t-1}$ are fixed polynomials of degree $0, \dots, t-1$ which have no common factors or roots representing elements of P.

Output: Each player i $(1 \leq i \leq n)$ determines $S_i \cap \mathrm{Rd}_{t-1}(S_1 \cup \dots \cup S_n)$.

1. Each player $i = 1, \dots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \dots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \dots, n$
   (a) receives the encryption of the polynomial λ_{i-1} from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} * f_i$ by utilizing the algorithm given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \prod_{i=1}^n f_i$ to players $2, \dots, c+1$
5. Each player $i = 1, \dots, c+1$
   (a) calculates the encryption of the $1, .., t-1$st derivatives of p, denoted $p^{(1)}, \dots, p^{(t-1)}$, by repeating the algorithm given in Sec. 4.1.2.
   (b) chooses random polynomials $r_{i,0}, \dots, r_{i,t-1} \leftarrow R^{nk}[x]$
   (c) calculates the encryption of the polynomial $\sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * r_{i,\ell}$ and sends it to all other players.
6. Each player $i = 1, \dots, n$
   (a) evaluates the encryption of the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=1}^{c+1} r_{i,\ell}\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.1.2, and sends them to all players
   (b) chooses a random element $b_{i,j,\ell}$ $(1 \leq j \leq n,\ 1 \leq \ell \leq k)$
   (c) for each ciphertext $c_{j\ell}$, calculate $b_{i,j,\ell} \times_h c_{j\ell}$ $(1 \leq j \leq n,\ 1 \leq \ell \leq k)$
7. The players i $(1 \leq i \leq n)$ calculate $U_{jm} = \left(\sum_{\ell=1}^n b_{\ell,j,m}\right) \times_h c_{jm}$ $(1 \leq j \leq n,\ 1 \leq m \leq k)$
8. All players decrypt the ciphertexts U_{ij}, so that only player i learns the decryption $a_{i,j}$.

For each player i $(1 \leq i \leq n)$, if $a_{i,j} = 0$ $(1 \leq j \leq k)$, then $(S_i)_j$ is in his result set.

Figure 4.6: Threshold Set-Union protocol secure against honest-but-curious adversaries (threshold contribution variant).

sabotages it. In group decryption schemes, players generally produce shares of the decrypted element; if one player sends a uniformly generated share instead of a valid one, the decrypted element is uniform. If the decrypted element is uniform, it conveys no information to the players. To ensure an encryption of an element in the threshold set is not decrypted once the element is known to be in the threshold set, a player sabotages the decryption under the following conditions: (1) he can decrypt the tag to $h(a) \mid\mid a$ for some a and (2) a has already been determined to be a member of the threshold set. All other ciphertexts should be correctly decrypted; either they are encryptions of elements in the threshold set which have not yet been decrypted, or they are encryptions of random elements.

Note that the protocol is the only protocol proposed in this chapter with a non-constant number of rounds. Because of the need to sabotage decryptions based on the results of past decryptions, there are $O(nk)$ rounds in this protocol.

Perfect Threshold Set-Union. This protocol is given in Fig. 4.7. Each player constructs the encrypted elements $(U_i)_j$ from the encrypted evaluation of $\Phi((S_i)_j)$ as written in step 6 of Figure 4.5. The players then utilize the Shuffle protocol to anonymously distribute these elements. If an element appears in the threshold set, then at least one encryption of it appears

Protocol: Threshold-Perfect-HBC
Input: There are $n \geq 2$ honest-but-curious players, $c < n$ dishonestly colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key for a homomorphic cryptosystem. The threshold number of repetitions at which an element appears in the output is t. F is a fixed polynomial of degree $t-1$ which has no roots representing elements of P. The threshold number of repetitions at which an element appears in the output is $t \geq 2$. $F_0, \ldots, F_{t-1}$ are fixed polynomials of degree $0, \ldots, t-1$ which have no common factors or roots representing elements of P. $\text{IsEq}(C, C') = 1$ if the ciphertexts C, C' encode the same plaintext, and 0 otherwise.
Output: Each player determines the elements of $\text{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Player 1 sends the encryption of the polynomial $\lambda_1 = f_1$ to player 2
3. Each player $i = 2, \ldots, n$
   (a) receives the encryption of the polynomial λ_{i-1} from player $i-1$
   (b) calculates the encryption of the polynomial $\lambda_i = \lambda_{i-1} * f_i$ by utilizing the algorithm given in Sec. 4.1.2.
   (c) sends the encryption of the polynomial λ_i to player $i+1 \mod n$
4. Player 1 distributes the encryption of the polynomial $p = \lambda_n = \prod_{i=1}^{n} f_i$ to players $2, \ldots, c+1$
5. Each player $i = 1, \ldots, c+1$
   (a) calculates the encryption of the $1, .., t-1$st derivatives of p, denoted $p^{(1)}, \ldots, p^{(t-1)}$, by repeating the algorithm given in Sec. 4.1.2.
   (b) chooses random polynomials $r_{i,0}, \ldots, r_{i,t-1} \leftarrow R^{nk}[x]$
   (c) calculates the encryption of the polynomial $\sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * r_{i,\ell}$ and sends it to all other players.
6. Each player $i = 1, \ldots, n$
   (a) evaluates the encryption of the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_\ell * \left(\sum_{i=1}^{c+1} r_{i,\ell}\right)$ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = \Phi((S_i)_j)$, using the algorithm given in Sec. 4.1.2, and sends them to all players
   (b) for each $i' = 1, \ldots, n$, $j = 1, \ldots, k$ chooses a random number $r_{i'j} \leftarrow R$ and calculates an encrypted element $U_{ij} = (r_{i'j} \times_h E_{pk}(c_{i'j}))$, and sends it to player i'
   (c) calculates the elements for $j = 1, \ldots, k$
   $U_{ij} = (r_{1j} \times_h E_{pk}(c_{1j})) +_h \ldots +_h (r_{nj} \times_h E_{pk}(c_{nj})) +_h E_{pk}((S_i)_j)$
   (d) constructs the set $V_i = \{U_{ij} \mid 1 \leq j \leq k\}$
7. By using the Shuffle protocol, all players perform shuffling on their private input sets V_i, obtaining the set U'.
8. For each shuffled ciphertext U'_ℓ with arbitrary ordering index $\ell \in [nk]$, the players $i = 1, \ldots, n$
   (a) each player i chooses random elements $q_{i,\ell} \leftarrow R$
   (b) calculate $W_\ell = U'_\ell +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_{i,\ell}\right)(\text{IsEq}(U'_\ell, U'_{\ell-1}) + \cdots + \text{IsEq}(U'_\ell, U'_1))\right)$
9. All players $1, \ldots, n$ decrypt each ciphertext W_ℓ, obtaining an element a_ℓ $(1 \leq \ell \leq nk)$.

If $a_j \in P$ $(1 \leq j \leq k)$, then a_j is a member of the result set.

Figure 4.7: Threshold Set-Union protocol secure against honest-but-curious adversaries (perfect variant).

in the shuffled ciphertexts. The players ensure in step 8 that all duplicates (ciphertexts of the same element) except the first have a random element added to them. This disguises the number of players who have each element of the threshold set in their private input. Let the shuffled ciphertexts U have an arbitrary ordering $U'_1, \ldots, U'_{nk}$. $\text{IsEq}(C, C') = 1$ if the ciphertexts C encode the same plaintext, and 0 otherwise. (This calculation can be achieved with the techniques in [36].) The players $i \in [n]$ then choose random elements $q_{i,\ell} \leftarrow R$ $(1 \leq \ell \leq nk)$ and decrypt the ciphertexts $W_\ell = U'_\ell +_h E_{pk}\left(\left(\sum_{i=1}^{n} q_\ell\right)(\text{IsEq}(U'_\ell, U'_{\ell-1}) + \ldots \text{IsEq}(U'_\ell, U'_1))\right)$. Thus, if U'_ℓ is a duplicate (encryption of an element which also appeared early in the ordering), it has a uniformly distributed element added to it, and conveys no information. Each element of the threshold set is decrypted exactly once, and all players thus learn the threshold set.

Security Analysis. We show that our protocol is correct, as each player learns the appropriate result set at its termination, and secure in the honest-but-curious model, as no player gains information that it would not gain when using its input in the ideal model. A formal statement of these properties is as follows:

Theorem 12. *In the Threshold Contribution Threshold Set-Union protocol of Fig. 4.6, every player i ($1 \leq i \leq n$) learns the set $S_i \cap Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Proof. Note that the encrypted computation is performed in accordance with Theorems 3 and 5, and thus the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. Each player i ($1 \leq i \leq n$) constructs encrypted evaluations of each $a \in S_i$, which are them homomorphically multiplied by a uniformly distributed element by all players. Thus, each ciphertext constructed in this fashion is either 0 (meaning $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$) or uniformly distributed (meaning $a \notin \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$). These ciphertexts are then decrypted; thus, each player i learns which elements of his private input appear in the threshold set $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. □

Theorem 13. *In the Semi-Perfect Threshold Set-Union protocol of Fig. 4.5, each player i ($1 \leq i \leq n$) learns the set $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Proof. Following the proof of Theorem 12, the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability and each shuffled element $T \mid\mid U$ is of one of the following forms:

- For some $a \in S_1 \cup \cdots \cup S_n$, $1 \leq i \leq n$, $T = Enc_i(h(a) \mid\mid a)$, U is of the form $E_{pk}(a)$ – thus, $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$
- For some $a \in S_1 \cup \cdots \cup S_n$, $1 \leq i \leq n$, $T = Enc_i(h(a) \mid\mid a)$, U is not of the form $E_{pk}(a)$ – thus, $a \notin \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$

The operation of Step 8 assures that for each $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, a corresponding U is correctly decrypted exactly once – all other decryptions of a are sabotaged to appear uniformly distributed. Thus, all players learn the elements of the set $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability. □

Theorem 14. *In the Perfect Threshold Set-Union protocol of Fig. 4.7, every player learns the set $Rd_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability.*

Proof. Following the proof of Theorem 12, the polynomial Φ is a polynomial representation of the multiset $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, with overwhelming probability and each shuffled (encrypted) element U'_ℓ ($1 \leq \ell \leq nk$) is of one of the following forms: $a \in P$ (indicating that $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$), or a uniformly distributed element (which can be distinguished from a representation of an element of P with overwhelming probability). Note that, if U'_ℓ is an encryption of an element a, and $\neg\exists_{\ell' \in [\ell-1]}\ U'_{\ell'}$ such that $U'_{\ell'}$ is also an encryption of a, then W_ℓ is also an encryption of a. (Otherwise, W_ℓ is an encryption of a uniformly distributed element.)

This calculation results in a list of encrypted elements W_ℓ, each of which is of one of the following forms: $a \in P$ (indicating that both: $a \in \mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$, and W_ℓ is with overwhelming probability the only encryption of a in the list), or a uniformly distributed element. Thus, when the players decrypt the list W_ℓ, they learn all elements of $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$ exactly once, with overwhelming probability. □

Theorem 15. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure and that the Shuffle protocol is secure, with overwhelming probability, in the Threshold Set-Union protocols of Figs. 4.5, 4.6, and 4.7, any coalition of fewer than n PPT honest-but-curious players learns no more information than would be gained by using the same private inputs in the ideal model with a trusted third party.*

Proof. Note that in the threshold contribution and perfect variants of Threshold Set-Union, all data is encrypted until the final result sets are revealed through joint decryption. As shown in Theorems 12 and 14, the final sets correspond exactly to the elements revealed (all elements that are not in the result set are uniformly distributed over R, and thus hold no information), no information except the result set is revealed to the players.

In the protocol for semi-perfect Threshold Set-Union, the result set is not decrypted all-at-once, but one element at a time. Theorem 13 shows the the resulting elements correspond exactly to the desired result set, but we must show that the behavior of each player during the process of decryption yields no disallowed information. Note that we require for the security of this protocol that a dishonest coalition hold no more than $t-1$ copies of any given element in their private input sets.

When performing the decryption process, each player learns two pieces of information when a result set element is revealed: the element, and whether the element revealed came from that player's own private input multiset. Each ciphertext is 'tagged', so each player can easily decide whether they constructed that ciphertext. Thus, if a dishonest coalition held at least t copies of any given element, they could determine that at least one other player also held a copy of that element, revealing forbidden information. However, as we have precluded this situation, no information is revealed; if a dishonest coalition holds $t-1$ copies of an element which appears in the result set, they already know that at least one other player holds it (otherwise it would not appear in the result set!). □

4.3.3 Malicious Case

By adding zero-knowledge proofs to our Over-Threshold Set-Union protocol secure against honest-but-curious adversaries, we extend our results to enable security against malicious adversaries. We provide details of our protocol secure against malicious adversaries in Section 4.4.4.

4.4 Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union for Malicious Parties

We extend the protocols for the Set-Intersection, Cardinality Set-Intersection, and Over-Threshold Set-Union problems given in Sections 4.2 and 4.3 to obtain security against adversaries in the malicious model. To obtain this result, we add zero-knowledge proofs, verified by all players, to ensure the correctness of all computation. In this section, we first introduce notation for zero-knowledge proofs, then give the protocols secure against malicious parties.

4.4.1 Tools

In this section, we describe cryptographic tools that we utilize in our protocols secure against malicious players.

Zero-Knowledge Proofs. We utilize several zero-knowledge proofs in our protocols for the malicious adversary model. We introduce the notation for these zero-knowledge proofs below; for additively homomorphic cryptosystems such as Paillier, we can efficiently construct these zero-knowledge proofs using standard constructions [9, 14].

- POPK$\{E_{pk}(x)\}$ denotes a zero-knowledge proof that given a public ciphertext $E_{pk}(x)$, the player knows the corresponding plaintext x [15].
- ZKPK$\{f \mid p' = f *_h \alpha\}$ is shorthand notation for a zero-knowledge proof of knowledge that the prover knows a polynomial f such that encrypted polynomial $p' = f *_h \alpha$, given the encrypted polynomials p' and α.
- ZKPK$\{f \mid (p' = f *_h \alpha) \wedge (y = E_{pk}(f))\}$ is the proof ZKPK$\{f \mid p' = f *_h \alpha\}$ with the additional constraint that $y = E_{pk}(f)$ (y is the encryption of f), given the encrypted polynomial p', y, and α.

Equivocal Commitment. A standard commitment scheme allows parties to give a "sealed envelope" that can be later opened to reveal exactly one value. We use an equivocal commitment scheme in our protocols secure against malicious players, such that the simulator can open the 'envelope' to an arbitrary value without being detected by the adversary [33, 39].

4.4.2 Set-Intersection Protocol for Malicious Adversaries

Our protocol for malicious parties performing Set-Intersection, given in Fig. 4.8, proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 4.2. The commitments to the data items $\Lambda(c_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs to prevent three forms of misbehavior: choosing ciphertexts for the encrypted coefficients of f_i without knowledge of their plaintext, not performing the polynomial multiplication of $f_j * r_{i,j}$ correctly, and not performing decryption correctly. We also constrain the leading coefficient of f_i to be 1 for all players, to prevent any player from setting their polynomial to 0; if $f_i = 0$, every element is a root, and thus it can represent an unlimited number of elements. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 4.2. The protocol can gain efficiency by taking advantage of the maximum coalition size c.

Our set-intersection protocol secure against malicious parties utilizes an expensive ($O(k^2)$ size) zero-knowledge proof to prevent malicious parties from cheating when multiplying the polynomial $r_{i,j}$ by the encryption of the polynomial f_j. Each player i must commit to each polynomial $r_{i,j}$ ($1 \leq i, j \leq n$), for purposes of constructing a zero-knowledge proof. We may easily replace this proof with use of the cut-and-choose technique, which requires only $O(k)$ communication.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world, where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 16. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, and the specified zero-knowledge proofs and proofs of correct decryption*

Protocol: Set-Intersection-Mal
Input: There are $n \geq 2$ players, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key to a homomorpic cryptosystem. The commitment scheme used in this protocol is a equivocal commitment scheme; each player holds any additional inputs necessary for this scheme, such as a common reference string.
Output: Each player determines $S_1 \cap \cdots \cap S_n$.

All players verify the correctness of all proofs sent to them, and stop participating in the protocol if any are not correct.

Each player $i = 1, \ldots, n$:

1. (a) calculates the polynomial f_i such that the k roots of the polynomial are the elements of S_i, as $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends δ_i, the encryption of the polynomial f_i to all other players along with proofs of plaintext knowledge for all coefficients except the leading coefficient (POPK$\{(\delta_i)_j\}$, $0 \leq j < k$).
   (c) for $1 \leq j \leq n$
      i. chooses a random polynomial $r_{i,j} \leftarrow R^k[x]$
      ii. sends a commitment to $\Lambda(r_{i,j})$ to all players, where $\Lambda(r_{i,j}) = E_{pk}(r_{i,j})$
2. for $1 \leq j \leq n$
   (a) opens the commitment to $\Lambda(r_{i,j})$
   (b) verifies proofs of plaintext knowledge for the encrypted coefficients of f_j
   (c) sets the leading encrypted coefficient (for x^k) to a known encryption of 1
   (d) calculates μ, the encryption of the polynomial $p_{i,j} = f_j * r_{i,j}$ with proofs of correct multiplication ZKPK$\{r_{i,j} \mid (\mu = r_{i,j} *_h \delta_j) \wedge (\Lambda(r_{i,j}) = E_{pk}(r_{i,j}))\}$ and sends it to all other players
3. All players
   (a) calculate the encryption of the polynomial $p = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j} = \sum_{i=1}^{n} f_i * (r_{j,i})$ as in Sec. 4.1.2, and verifies all attached proofs
   (b) perform a group decryption to obtain the polynomial p, and distribute proofs of correct decryption

Each player $i = 1, \ldots, n$ determines the intersection multiset as follows: for each $a \in S_i$, he calculates b such that $(x - a)^b | p \ \wedge \ (x - a)^{b+1} \not| p$. The element a appears b times in the intersection multiset.

Figure 4.8: Set-Intersection protocol secure against malicious adversaries.

cannot be forged, then in the Set-Intersection protocol secure against malicious adversaries in Fig. 4.8, for any coalition Γ of colluding players (at most $n - 1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and Γ in the real model.

Proof. In this simulation proof, we give an algorithm for a player G in the ideal model. This player communicates with the malicious players Γ, pretending to be one or more honest players in such a fashion that Γ cannot distinguish that he is not in the real world. We assume that all malicious players can collude. The trusted third party takes the input from G and the honest parties, and gives both G and the honest parties the intersection set. G then communicates with the malicious players Γ, so they also learn the intersection set. A graphical representation of these players is given in Figure 4.9

We give a sketch of how the player G operates (note that G can prevaricate when opening commitments, as we use an equivocal commitment scheme, and can extract plaintext from proofs of plaintext knowledge):

1. For each simulated honest player i, G:

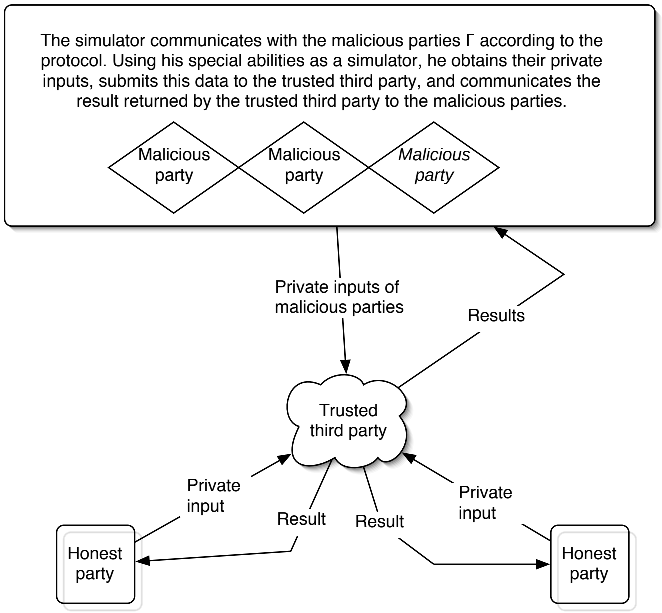

The simulator communicates with the malicious parties Γ according to the protocol. Using his special abilities as a simulator, he obtains their private inputs, submits this data to the trusted third party, and communicates the result returned by the trusted third party to the malicious parties.
Malicious party
Malicious party
Malicious party
Private inputs of malicious parties
Results
Trusted third party
Private input
Result
Result
Private input
Honest party
Honest party

Figure 4.9: A simulation proof defines the behavior of the player G, who translates between the malicious players Γ, who believe they are operating in the real model, and the ideal model, in which the trusted third party computes the desired answer.

   (a) chooses a polynomial f_i such that each such polynomial is relatively prime and has leading coefficient 1 (for randomly generated polynomials with leading coefficient 1, this is true with overwhelming probability)
   (b) chooses arbitrary polynomials $r_{i,1}, \ldots, r_{i,n}$ and creates encryptions $\Lambda(r_{i,j})$ from them (in the case of Paillier, specially construct encryptions of those polynomials, and proofs of knowledge of each coefficient, see Section 4.4.1)

2. Performs step 1 of the protocol:
   (a) sends the encryption of f_i to all malicious players Γ, along with proofs of plaintext knowledge and commitments to $\Lambda(r_{i,j})$ $(1 \leq j \leq n)$
   (b) sends data items $\Lambda(r_{i,j})$ $(1 \leq j \leq n)$ to all malicious players Γ
   (c) Receives from each malicious player $\alpha \in \Gamma$:
      i. encryption of a polynomial f_α and proofs of plaintext knowledge for its coefficients
      ii. trapdoor commitments to data items $\Lambda(r_{\alpha,j})$ for each random polynomial $r_{\alpha,j}$, $1 \leq j \leq n$
3. The player G extracts from the proofs of plaintext knowledge and trapdoor commitments to $\Lambda(r_{i,j})$ (in the case of Paillier, the extraction is from the proof of knowledge of the discrete logarithm), the polynomials f_α, and the random polynomials $r_{\alpha,j}$ the malicious players Γ have chosen.
4. G obtains the roots of each polynomial f_α (as these exactly determine, for the purposes of the protocol, his set):
   - If polynomial factoring is possible, G may factor f_α. $f_\alpha(a) = 0 \Leftrightarrow (x-a)|f_\alpha$, so all roots of f_α may be determined by examining the linear factors.

- If we are working in the random oracle model, then, with overwhelming probability, to correctly represent any element of the valid set P, a player must consult the random oracle. As there can be only a polynomial number of such queries, for each query a, G may check if $f_\alpha(a \mid\mid h(a)) = 0$.
- If neither of these routes are feasible, then a proof that f_α was constructed by multiplying k linear factors of the form $x - a$ may be added to the protocol instead of proofs of plaintext knowledge. This proof is of size $O(k^3)$, and is constructed by using proofs of plaintext knowledge for some linear factors, and layering proofs of correct multiplication to obtain the complete polynomial f_α. From this proof, each linear factor of f_α can be obtained, and thus all roots of f_α.

5. G submits the sets represented by these roots to the trusted third party. The honest players submit their private input sets to the trusted third party. The trusted third party returns the intersection set I to G and the honest players.
6. G prepares to reveal the intersection set to the malicious players Γ:
   (a) selects a target polynomial $p = \left(\prod_{a \in I}(x - a)\right) * s$, where s is chosen uniformly from those polynomials of degree $2k - |I|$. (note that, by Lemma 2, this is exactly the polynomial calculated by simply running the protocol, as by the reasoning of Lemma 19, all coefficients of f_i $(1 \leq i \leq n)$ are in the set $R^* \cup \{0\}$.)
   (b) chooses a set of polynomials $r_{i,j}$ (where i is one of the simulated honest players) such that $\sum_{i=1}^{n} f_i \left(\sum_{j=1}^{n} r_{i,j}\right) = p$ (from the proof of Lemma 2, we know that such polynomials exist, and can be determined through simple polynomial manipulation)
7. G follows the rest of the protocol with the malicious players Γ as written, except that he opens the trapdoor commitment to reveal an appropriate $\Lambda(r_{i,j})$ for the new chosen $r_{i,j}$. In this way, the players calculate an encryption of the polynomial p chosen by G, and then decrypt it. The coalition players thus learn the intersection set.

Note that the dishonest players cannot distinguish that they are talking to G (who is working in the ideal model) instead of other clients (in the real world), and the correct answer is learned by all parties, in both the real and ideal models.

□

4.4.3 Cardinality Set-Intersection Protocol for Malicious Adversaries

We give a protocol, secure against malicious parties, to perform Cardinality Set-Intersection in Fig. 4.10. It proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 4.3. The commitments to the data items $\Lambda(r_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs of knowledge to prevent five forms of misbehavior: choosing f_i without knowledge of its roots, choosing f_i such that it is not the product of linear factors, not performing the polynomial multiplication of $f_j * r_{i,j}$ correctly, not calculating encrypted elements $(V_i)_j$ correctly (either not from the data items $(S_i)_j$ or not evaluating the encrypted polynomial p), and not performing decryption correctly. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 4.3.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world,

Protocol: Cardinality-Mal
Input: There are $n \geq 2$ players, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key to a homomorpic cryptosystem. The commitment scheme used in this protocol is a equivocal commitment scheme.
Output: Each player determines $|S_1 \cap \cdots \cap S_n|$.

All players verify the correctness of all proofs sent to them, and stop participating in the protocol if any are not correct.

Each player $i = 1, \ldots, n$:

1. (a) calculates the polynomial f_i such that the k roots of the polynomial are the elements of S_i, as $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
   (b) sends:
      i. encrypted elements $y_{i,1} = E_{pk}((S_i)_1), \ldots, y_{i,k} = E_{pk}((S_i)_k)$ to all other players, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{i,j})\}$, $1 \leq j < k$)
      ii. sends δ_i, the encryption of the polynomial f_i to all other players, along with a proof of correct construction $\text{ZKPK}\left\{ a_1, \ldots, a_k \;\middle|\; \begin{array}{l} \tau_i = ((x - a_1) *_h \ \ldots \ *_h \ (x - a_{k-1}) *_h \ \alpha) \\ \wedge \quad y_{i,1} = E_{pk}(a_1) \wedge \cdots \wedge y_{i,k} = E_{pk}(a_k) \\ \wedge \quad \alpha = E_{pk}(x - a_k) \end{array} \right\}$
   (c) for $1 \leq j \leq n$
      i. chooses a random polynomial $r_{i,j} \leftarrow R^k[x]$
      ii. sends a commitment to $\Lambda(r_{i,j})$ to all players, where $\Lambda(r_{i,j}) = E_{pk}(r_{i,j})$
2. for $1 \leq j \leq n$
   (a) opens the commitment to $\Lambda(r_{i,j})$
   (b) verifies proofs of plaintext knowledge for the encrypted coefficients of f_j
   (c) sets the leading encrypted coefficient (for x^k) to a known encryption of 1
   (d) calculates $\tau_{i,j}$, the encryption of the polynomial $p_{i,j} = f_j * r_{i,j}$, with proofs of correct multiplication ZKPK$\{r_{i,j} \mid (\tau_{i,j} = r_{i,j} *_h \delta_j) \wedge (\Lambda(r_{i,j}) = E_{pk}(r_{i,j}))\}$ and sends it to all other players
3. Each player $i = 1, \ldots, n$:
   (a) calculates μ, the encryption of the polynomial $p = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{i,j}$, as in Sec. 4.1.2, and verifies all attached proofs
   (b) evaluates the encryption of the polynomial p at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.1.2.
   (c) for each $j \in [k]$ chooses a random element r_{ij}, calculates an encrypted element $(V_i)_j = r_{ij} \times_h E_{pk}(c_{ij})$, with attached proof of correct construction ZKPK$\{(r_{ij}, z) \mid ((V_i)_j = r_{ij} \times_h \mu(z)) \wedge (y_{i,j} = E_{pk}(z))\}$, and sends the encrypted element $(V_i)_j$ and the proof of correct construction to all players
4. All players perform the Shuffle protocol on the sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized.
5. All players $1, \ldots, n$ decrypt each element of the shuffled set V (and send proofs of correct decryption to all other players)

If nb of the decrypted elements from V are 0, then the size of the set intersection is b.

Figure 4.10: Cardinality set-intersection protocol secure against malicious adversaries.

where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 17. *Assuming that the additively homomorphic, threshold cryptosystem $E_{pk}(\cdot)$ is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Cardinality Set-Intersection protocol secure against malicious adversaries in Fig. 4.10, for any coalition Γ of colluding players (at most $n-1$ such colluding parties), there is a player (or group of players) G operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable* *from the views of the honest players and* Γ *in the real model.*

Proof. The simulation proof of this theorem follows the proof of Theorem 16 with only small changes; the additional zero-knowledge proofs in the protocol are generally irrelevant to the operation of the simulator. □

4.4.4 Over-Threshold Set-Union Protocol for Malicious Adversaries

We give a protocol, secure against malicious parties, to perform Over-Threshold Set-Union in Fig. 4.11. It proceeds largely as the protocol secure against honest-but-curious parties, which was given in Fig. 4.4. The commitments to the data items $\Lambda(r_{i,j})$ are purely for the purposes of a simulation proof. We add zero-knowledge proofs of knowledge to prevent six forms of misbehavior: choosing f_i without knowledge of its roots, choosing f_i such that it is not the product of linear factors, not performing the polynomial multiplication of $f_j * \lambda_{j-1}$ correctly, not calculating $\alpha_{i,\ell} = p^{(\ell)} * r_{i,\ell}$ $(0 \leq \ell \leq t-1)$ correctly, not calculating encrypted elements $(V_i)_j$ correctly (either not from the data items $(S_i)_j$ or not evaluating the encrypted polynomial Φ), and not performing decryption correctly. We can thus detect or prevent misbehavior from malicious players, forcing this protocol to operate like the honest-but-curious protocol in Fig. 4.4.

Security Analysis. We provide a simulation proof of this protocol's security; an intermediary G translates between the real wold with malicious, colluding PPT players Γ and the ideal world, where a trusted third party computes the answer set. Our proof shows that no Γ can distinguish between the ideal world and the real world, thus no information other than that in the answer set can be gained by malicious players. A formal statement of our security property is as follows:

Theorem 18. *Assuming that the additively homomorphic, threshold cryptosystem* $E_{pk}(\cdot)$ *is semantically secure, the Shuffle protocol is secure, and the specified zero-knowledge proofs and proofs of correct decryption cannot be forged, then in the Over-Threshold Set-Union protocol secure against malicious adversaries in Fig. 4.10, for any coalition* Γ *of colluding players (at most* $n-1$ *such colluding parties), there is a player (or group of players)* G *operating in the ideal model, such that the views of the players in the ideal model is computationally indistinguishable from the views of the honest players and* Γ *in the real model.*

Proof. In this simulation proof, we give an algorithm for a player G in the ideal model. This player communicates with the malicious players Γ, pretending to be one or more honest players in such a fashion that Γ cannot distinguish that he is not in the real world. We assume that all malicious players can collude. The trusted third party takes the input from G and the honest parties, and gives both G and the honest parties the intersection set. G then communicates with the malicious players Γ, so they also learn the intersection set. A graphical representation of these players is given in Figure 4.9.

We give a sketch of how the player G operates (note that G can prevaricate when opening commitments, as we use an equivocal commitment scheme, and can extract plaintext from proofs of plaintext knowledge):

1. For each simulated honest player i, G:
   (a) chooses a set S'_i of arbitrary elements $(S'_i)_1, \ldots, (S'_i)_k \in R$

Protocol: OverThreshold-Mal

Input: There are $n \geq 2$ players, $c < n$ maliciously colluding, each with a private input set S_i, such that $|S_i| = k$. The players share the secret key sk, to which pk is the corresponding public key to a homomorpic cryptosystem. The commitment scheme used in this protocol is a equivocal commitment scheme. The threshold number of repetitions at which an element appears in the output is t. $F_0, \ldots, F_{t-1}$ are fixed polynomials of degree $0, \ldots, t-1$ which have no common factors or roots representing elements of P.

Output: Each player determines $\mathrm{Rd}_{t-1}(S_1 \cup \cdots \cup S_n)$.

All players verify the correctness of all proofs sent to them, and refuse to participate in the protocol if any are not correct.

Each player $i = 1, \ldots, n$:

1. Each player $i = 1, \ldots, n$ calculates the polynomial $f_i = (x - (S_i)_1) \ldots (x - (S_i)_k)$
2. Players $1, \ldots, c+1$ send commitments to $y_{i,1}, \ldots, y_{i,k}$ to all players, where $y_{i,j} = E_{pk}((S_i)_j)$ $(1 \leq j \leq k)$. All players then open these commitments.
3. Player 1 sends to all other players: encrypted elements $y_{1,1} = E_{pk}((S_1)_1), \ldots, y_{1,k} = E_{pk}((S_1)_k)$, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{1,j})\}$, $1 \leq j < k$); τ_1, the encryption of the polynomial $\lambda_1 = f_1$ to all other players, along with a proof of correct construction $$\mathrm{ZKPK}\left\{ a_1, \ldots, a_k \;\middle|\; \begin{array}{rl} & \tau_1 = ((x - a_1) *_h \ \ldots \ *_h \ (x - a_{k-1}) *_h \ \alpha) \\ \wedge & y_{1,1} = E_{pk}(a_1) \wedge \cdots \wedge y_{1,k} = E_{pk}(a_k) \\ \wedge & \alpha = E_{pk}(x - a_k) \end{array} \right\}$$
4. Each player $i = 2, \ldots, n$
   (a) receives τ_i, the encryption of the polynomial λ_{i-1}, from player $i - 1$
   (b) sends to all other players: encrypted elements $y_{i,1} = E_{pk}((S_i)_1), \ldots, y_{i,k} = E_{pk}((S_i)_k)$, along with proofs of plaintext knowledge (POPK$\{E_{pk}(y_{i,j})\}$, $1 \leq j < k$); τ_i, the encryption of the polynomial $\lambda_i = f_i * \lambda_{i-1}$, along with a proof of correct construction $$\mathrm{ZKPK}\left\{ a_1, \ldots, a_k \;\middle|\; \begin{array}{rl} & \tau_i = ((x - a_1) *_h \ \ldots \ *_h \ (x - a_k) *_h \ \tau_{i-1}) \\ \wedge & y_{i,1} = E_{pk}(a_1) \wedge \cdots \wedge y_{i,k} = E_{pk}(a_k) \end{array} \right\}$$
5. Each player $i = 1, \ldots, c + 1$
   (a) choose random polynomials $r_{i,0}, \ldots, r_{i,t-1} \leftarrow R^k[x]$
   (b) for $\ell = 0, \ldots, t-1$, calculate α_ℓ the encryption of the ℓth derivative of $p = \lambda_n$, denoted $p^{(\ell)}$, by repeating the algorithm given in Sec. 4.1.2.
   (c) calculate $\alpha_{i,\ell}$, the encryption of the polynomial $p^\ell * r_{i,\ell}$, for $0 \leq \ell \leq t - 1$ and send them to all other players, along with proofs of correct polynomial multiplication, $\mathrm{ZKPK}\left\{ r_{i,\ell} \;\middle|\; \alpha_{i,\ell} = r_{i,\ell} *_h \ p^{(\ell)} \right\}$
6. Each player $i = 1, \ldots, n$:
   (a) calculates μ, the encryption of the polynomial $\Phi = \sum_{\ell=0}^{t-1} p^{(\ell)} * F_j * \left(\sum_{i=0}^{c+1} r_{i,\ell}\right)$, as in Sec. 4.1.2, and verifies all attached proofs
   (b) evaluates the encryption of the polynomial Φ at each input $(S_i)_j$, obtaining encrypted elements $E_{pk}(c_{ij})$ where $c_{ij} = p((S_i)_j)$, using the algorithm given in Sec. 4.1.2.
   (c) for each $j \in [k]$ chooses a random element $r_{ij} \leftarrow R$, calculates an encrypted element $(V_i)_j = (r_{ij} \times_h E_{pk}(c_{ij})) + (S_i)_j$, with attached proof of correct construction $\mathrm{ZKPK}\{(r_{ij}, z) \mid ((V_i)_j = (r_{ij} \times_h \mu(z)) + z) \wedge (y_{i,j} = E_{pk}(z))\}$, and sends the encrypted element $(V_i)_j$ and the proof of correct construction to all players
7. All players perform the Shuffle protocol on the sets V_i, obtaining a joint set V, in which all ciphertexts have been re-randomized, then jointly decrypt each element of the shuffled set V (and send proofs of correct decryption to all other players).

Each element $a \in P$ that appears b times in V is an element in the threshold set that appears b times in the players' private inputs.

Figure 4.11: Over-threshold set-intersection protocol secure against malicious adversaries.

   (b) Performs steps $1 - 2$ of the protocol, sending equivocal commitments to the set S_i for each simulated honest player.

2. The player G extracts the private input sets chosen by Γ, for each malicious player, from the equivocal commitments sent in step 2 of the protocol. G submits the sets extracted from these commitments to the trusted third party. The honest players submit their

private input sets to the trusted third party. The trusted third party returns the result set I to G and the honest players.

3. G prepares to reveal the intersection set to the malicious players Γ: G chooses new sets S_i to replace the sets S_i' used to construct the commitment. These sets are chosen to contain the following elements:
   (a) for each element a that appears $b > 0$ in I, and b_Γ times in the private input multisets of the malicious players (Γ), the element a is included $b + t - 1 - b_\Gamma$ times in the multisets S_i
   (b) all elements not specified by the prior rule are chosen uniformly from R
4. G follows the rest of the protocol with the malicious players Γ as written. The coalition players thus learn the result set.

Note that the dishonest players cannot distinguish that they are talking to G (who is working in the ideal model) instead of other clients (in the real world), and the correct answer is learned by all parties, in both the real and ideal models.

□

4.5 Other Applications of Our Multiset Computation Techniques

Our techniques for privacy-preserving computation of multiset operations have wide applicability beyond the protocols we discuss in Sections 4.2 and 4.3. We first discuss the composition of our techniques to compute arbitrary functions based on the intersection, union, and reduction operators. We also propose an efficient method for the Subset problem, determining whether $A \subseteq B$. As an example of the application of our techniques to problems outside the realm of set computation, we describe their use in evaluation of boolean formulae.

4.5.1 General Computation on Multisets

Our techniques for privacy-preserving multiset operations can be arbitrarily composed to enable a wide range of privacy-preserving multiset computations. In particular, we give a grammar describing functions on multisets that can be efficiently computed using our privacy-preserving operations:

$$\Upsilon ::= s \mid \mathrm{Rd}_d(\Upsilon) \mid \Upsilon \cap \Upsilon \mid s \cup \Upsilon \mid \Upsilon \cup s,$$

where s represents any multiset held by some player, and $d \geq 1$. Note that any monotone function on multisets can be expressed using the grammar above, and thus our techniques for privacy-preserving multiset operations are truly general.

It is worth noting that the above grammar only allows computation of the union operator when at least one of the two operands is a multiset known to some player. Although any monotone function on multisets can be described by our grammar, in some cases it is desirable (or more efficient) to enable the calculation of the union operator on two multisets calculated from other multiset operations, such that neither operand is known to any player. In this case, we could calculate the union operation in the following way. Let λ and $E_{pk}(f)$ be the encrypted polynomial representations of the two multisets. The players use standard techniques to privately obtain additive shares $f_1, \ldots, f_\nu$ of f, given $E_{pk}(f)$. Using these shares, they then calculate $(f_1 *_h \lambda) +_h \ldots +_h (f_\nu *_h \lambda) = f *_h \lambda$, the encryption of the polynomial representation of the union multiset.

4.5.2 Private Subset Relation

Problem Statement Let the set A be held by Alice. The set B may be the result of an arbitrary function over multiple players' input sets (for example as calculated using the grammar above). The *Subset* problem is to determine whether $A \subseteq B$ without revealing any additional information.

Let λ be the encryption of the polynomial p representing B. Note that $A \subseteq B \Leftrightarrow \forall_{a \in A}\, p(a) = 0$. Alice thus evaluates the encrypted polynomial λ at each element $a \in A$, homomorphically multiplies a random element by each encrypted evaluation, and adds these blinded ciphertexts to obtain β'. If β' is an encryption of 0, then $A \subseteq B$. More formally:

1. For each element $a = A_j$ $(1 \leq j \leq |A|)$, the player holding A:
   (a) calculates $\beta_j = \lambda(a)$
   (b) chooses a random element $b_j \leftarrow R$, and calculates $\beta'_j = b_j \times_h \beta_j$
2. The player holding A calculates $\beta' = \beta'_1 +_h \ldots +_h \beta'_{|A|}$
3. All players together decrypt β' to obtain y. If $y = 0$, then $A \subseteq B$.

This protocol may be simply extended to allow the set A to be held by multiple players, such that $A = A_1 \cup \cdots \cup A_\nu$, where each set A_i is held by a single player.

4.5.3 Computation of CNF Formulae

Finally, we show that our techniques on private multiset operations have applications outside of the realm of multiset computations. As a concrete example, we show that we can apply our techniques to efficient privacy-preserving evaluation of boolean formulae, in particular, the conjunctive normal form (CNF). A formula in CNF is a conjunction of a number of disjunctive clauses, each of which is formed of several variables (or their negations).

Problem Statement Let ϕ be a public CNF boolean formula on variables $V_1, \ldots, V_\kappa$. Each player knows the truth assignment to some subset of $\{V_1, \ldots, V_\kappa\}$, where each variable is known to at least one player. The players cooperatively calculate the truth value of ϕ under this assignment, without revealing any other information about the variable assignment.

We address this problem by introducing multiset representations of boolean formulae. Let True and False be distinct elements of R (e.g., 0 and 1). For each variable in the formula, let the multiset representation of the variable be { True} if its value is true, and { False} if its value is false. Then, replace each $\vee$ operator in ϕ with a $\cup$ operator, and each $\wedge$ operator with a $\cap$ operator. If True is a member of the resulting multiset, then ϕ is true. The polynomial multiset representation of the CNF formula can now be evaluated by the players through use of our privacy-preserving multiset operations, as the function is described in the grammar given in Section 4.5.1.

We can also solve many variations on boolean formula evaluation using our techniques. For example, we might require, instead of using the boolean operations, that at least t of the variables in a clause be satisfied. Note that using our techniques can be more efficient than standard multiparty techniques, as standard techniques require an expensive multiplication operation, involving all players, to compute the $\wedge$ operator [5, 28].

4.6 Proof of Mathematical Lemmas

In this section, we prove Lemmas 2 and 4, as well as several lemmas on which these proofs depend.

4.6.1 Proof of Lemma 2

Lemma 19. *Let the ring R have subfields $\mathcal{F}_1$ and $\mathcal{F}_2$. Define the* gcd *of two polynomials $f, g \in R[x]$ as the output of Euclid's algorithm for computing the greatest common denominator; if Euclid's algorithm fails, the* gcd *is undefined.*
Any PPT adversary who can obtain (with non-negligible probability) two polynomials for which the gcd *is undefined can determine the size of the subfields of R (with non-negligible probability).*

Proof. If the leading coefficient of a polynomial p is in R^*, then for any polynomial $b \in R[x]$, there exist unique polynomials q, r such that $p = q * b + r$ $(\deg(r) < \deg(b))$ [51]. Note that this is the sole calculation necessary to compute the Euclidean gcd algorithm, and that this algorithm runs in PPT. Thus, if this algorithm fails to compute $\gcd(f, g)$, it must have calculated some polynomial p' as an intermediate result such that the leading coefficient of p' is in $R \setminus (R^* \cup \{0\})$. The elements of $R \setminus (R^* \cup \{0\})$ are those without a multiplicitive inverse: multiples of $|\mathcal{F}_\ell|$ $(1 \leq \ell \leq 2)$. Thus, the polynomial p that causes Euclid's algorithm to fail must have a leading coefficient of a multiple of the size of some sub-field $\mathcal{F}_\ell$ $(1 \leq \ell \leq 2)$. Given this coefficient, one can compute $|\mathcal{F}_1|, |\mathcal{F}_2|$ in probabilistic polynomial time by using the Euclidean algorithm over integers. As, by assumption in Section 3.2.1, this problem is hard over our ring R, we can (with overwhelming probability) compute $\gcd(f, g)$ using Euclid's algorithm. □

Remark 20. *For all $a, b \in R$, if $a \in R^*$, there exists no element $c \in R$ $(c \neq b)$ such that $ab = ac$.*

Lemma 21. *For all polynomials $f, g \in R[x]$ such that the leading coefficient of f is a member of R^*, there exists no polynomial $y \in R[x]$ such that $f * g = f * y$, $g \neq y$.*

Proof. For two polynomials to be equal, each of their coefficients must be equal. Thus, we may express the condition $f * g = f * y$ as follows for each $\ell \geq 0$:

$$\begin{aligned}
(f * g)[\ell] &= \sum_{j=0}^{\ell} f[\ell - j] g[j] \\
(f * y)[\ell] &= \sum_{j=0}^{\ell} f[\ell - j] y[j] \\
\sum_{j=0}^{\ell} f[\ell - j] \left(g[j] - y[j] \right) &= 0
\end{aligned}$$

We prove by induction that $g[\ell] = y[\ell]$ $(0 \leq \ell \leq \deg(g))$. As a base case, we prove that $g[\deg(g)] = y[\deg(g)]$:

$$\begin{aligned} \sum_{j=0}^{\deg(f)+\deg(g)} f[\deg(f) + \deg(g) - j](g[j] - y[j]) &= 0 \\ f[\deg(f)](g[\deg(g)] - y[\deg(g)]) &= 0 \end{aligned}$$

Because $f[\deg(f)] \in R^*$ (by definition, $f[\deg(f)] \neq 0$), by Lemma 20 $g[\deg(g)] = y[\deg(g)]$.

We now make the strong inductive assumption that for $i \leq \ell \leq \deg(g)$, $g[\ell] = y[\ell]$. Next, we use this assumption to prove that $g[i-1] = y[i-1]$:

$$\begin{aligned} \sum_{j=0}^{\deg(f)+i-1} f[\deg(f) + i - 1 - j]\,(g[j] - y[j]) &= 0 \\ \sum_{j=0}^{\deg(f)+i-1} f[\deg(f) + i - 1 - j]\,(g[j] - y[j]) &= f[\deg(f)]\,(g[i-1] - y[i-1]) \\ f[\deg(f)]\,(g[i-1] - y[i-1]) &= 0 \end{aligned}$$

Because $f[\deg(f)] \in R^*$ (by definition, $f[\deg(f)] \neq 0$), by Lemma 20 $g[i-1] = y[i-1]$. Thus, by the inductive principle, all coefficients of g are identical to those of y up to $\deg(g)$. We now prove that $\deg(y) \leq \deg(g)$, showing that $y = g$ (and thus that our lemma is true).

If $\deg(y) > \deg(g)$ then $\exists_{\ell > \deg(g)}\ y[\ell] \neq 0$ (let ℓ be the minimal such index):

$$\begin{aligned} g[\ell] &= 0 \\ g[\ell] - y[\ell] &\neq 0 \\ \sum_{j=0}^{i} f[i-j]\,(g[j] - y[j]) &= 0 \end{aligned}$$

We may remove from the sum all terms for which $g[j] - y[j] = 0$, leaving us with the following equation for some $i \leq \deg(f)$:

$$f[\deg(f)]\,(g[\ell] - y[\ell]) = 0$$

Because $f[\deg(f)] \in R^*$ (by definition, $f[\deg(f)] \neq 0$), by Lemma 20 $y[\ell] = g[\ell] = 0$. Thus, no such index ℓ can exist; $\deg(y) \leq \deg(g)$. Because we also know that all terms of y up to $\deg(g)$ are identical to those of g, we may conclude that $y = g$, and thus that our lemma is true. □

Lemma 22. *For all polynomials $f_1, f_2, g_1, g_2 \in R[x]$ such that $f_1 * g_1 = f_2 * g_2$, $\gcd(f_1, f_2) = 1$, then $f_2 \mid g_1$.*

Proof. We defined $\gcd(f_1, f_2)$ with $f_1, f_2 \in R[x]$ as the output of Euclid's algorithm for calculating the gcd (which succeeds with overwhelming probability by Lemma 19). Note that from the intermediate results of this calculation we can determine polynomials $p_1, p_2 \in R[x]$ such that $p_1 * f_1 + p_2 * f_2 = 1$, as $\gcd(f_1, f_2) = 1$.

$$\begin{aligned}
p_1 * f_1 + p_2 * f_2 &= 1 \\
p_1 * f_1 &= 1 - p_2 * f_2 \\
g_1 * p_1 * f_1 &= g_1(1 - p_2 * f_2) \\
p_1 * (f_2 * g_2) &= g_1(1 - p_2 * f_2) \\
g_1 &= f_2\,(p_1 * g_2 + p_2 * g_1)
\end{aligned}$$

Because there exists a polynomial $p_1 * g_2 + p_2 * g_1 \in R[x]$ such that $g_1 = f_2\,(p_1 * g_2 + p_2 * g_1)$, $f_2 \mid g_1$. □

Lemma 2. *Let f, g be polynomials in $R[x]$ where R is a ring such that no PPT adversary can find the size of its subfields with non-negligible probability,* $\deg(f) = \deg(g) = \alpha$*,* $\beta \geq \alpha$*,* $\gcd(f, g) = 1$*, and* $f[\deg(f)] \in R^* \wedge g[\deg(g)] \in R^*$*. Let $r = \sum_{i=0}^{\beta} r[i]x^i$ and $s = \sum_{i=0}^{\beta} s[i]x^i$, where $\forall_{0 \leq i \leq \beta}\ r[i] \leftarrow R$, $\forall_{0 \leq i \leq \beta}\ s[i] \leftarrow R$ (independently).*

Let $u = f * r + g * s = \sum_{i=0}^{\alpha+\beta} u[i]x^i$. Then $\forall_{0 \leq i \leq \alpha+\beta}\ u[i]$ are distributed uniformly and independently over R.

Proof. For clarity, we give a brief outline of the proof before proceeding to the details. Given any fixed polynomials f, g, u, we calculate the number z of r, s pairs such that $f * r + g * s = u$. We may then check that, given any fixed polynomials f, g, the total number of possible r, s pairs, divided by z, is equal to the number of possible result polynomials u. This implies that, if $\gcd(f, g) = 1$ and we choose the coefficients of r, s uniformly and independently from R, the coefficients of the result polynomial u are distributed uniformly and independently over R.

We now determine the value of z, the number of r, s pairs such that $f * r + g * s = u$. Let us assume that for this particular u there exists at least one pair $\hat{r}, \hat{s}$ such that $f * \hat{r} + g * \hat{s} = u$. For any pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, then

$$\begin{aligned}
f * \hat{r} + g * \hat{s} &= f * \hat{r}' + g * \hat{s}' \\
f * (\hat{r} - \hat{r}') &= g * (\hat{s}' - \hat{s})
\end{aligned}$$

As $\gcd(f, g) = 1$, we may conclude that $g|(\hat{r} - \hat{r}')$ and $f|(\hat{s}' - \hat{s})$ by Lemma 22. Let $p * g = \hat{r} - \hat{r}'$ and $p * f = \hat{s}' - \hat{s}$.

We must now show that each polynomial p, of degree at most $\beta - \alpha$, determines exactly one unique pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, and that there exist no pairs $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$ that are not generated by a single choice of the polynomial p of degree at most $\beta - \alpha$.

To show that there exist no pairs $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$ that are not generated by some choice of the polynomial p, of degree at most $\beta - \alpha$, we let $p' * g = \hat{r} - \hat{r}'$ and $p * f = \hat{s}' - \hat{s}$ for any p', p of degree at most $\beta - \alpha$. As we proved that $g|(\hat{r} - \hat{r}')$ and $f|(\hat{s}' - \hat{s})$, we can represent f and g in this fashion without loss of generality.

$$\begin{aligned}
f * (\hat{r} - \hat{r}') &= g * (\hat{s}' - \hat{s}) \\
f * (p' * g) &= g * (p * f)
\end{aligned}$$

As the leading coefficients of f and g are members of $(R^* \cup \{0\})$, we may apply Lemma 21 to remove both f and g from our equation, leaving the fact that $p = p'$. Thus, there exist no pairs

$\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$ that are not generated by some choice of the polynomial p, of degree at most $\beta - \alpha$.

To show that each polynomial p, of degree at most $\beta - \alpha$, determines exactly one unique pair $\hat{r}', \hat{s}'$ such that $f * \hat{r}' + g * \hat{s}' = u$, note that $\hat{r}' = \hat{r} - g * p$, $\hat{s}' = \hat{s} + f * p$; as we have fixed $f, g, \hat{r}, \hat{s}$, a choice of p determines both $\hat{r}', \hat{s}'$. If these assignments were not unique, there would exist polynomials p, p' such that either $\hat{r}' = \hat{r} - g * p = \hat{r} - g * p'$ or $\hat{s}' = \hat{s} + f * p = \hat{s} + f * p'$. These conditions imply that either $g * p = g * p'$ or $f * p = f * p'$ for some polynomials $p \neq p'$; we know this is impossible (when the leading coefficients of f and g are members of $R^* \cup \{0\}$) by Lemma 21.

Thus the number of polynomials p, of degree at most $\beta - \alpha$, is exactly equivalent to the number of r, s pairs such that $f * r + g * s = u$. As there are $|R|^{\beta-\alpha+1}$ such polynomials p, $z = |R|^{\beta-\alpha+1}$.

We now show that the total number of r, s pairs, divided by z, is equal to the number of result polynomials u. There are $|R|^{2\beta+2}$ r, s pairs. As $\frac{|R|^{2\beta+2}}{z} = \frac{|R|^{2\beta+2}}{|R|^{\beta-\alpha+1}} = |R|^{\alpha+\beta+1}$, and there are $|R|^{\alpha+\beta+1}$ possible result polynomials, we have proved the theorem true. □

4.6.2 Proof of Lemma 4

Lemma 23. *For all polynomials $q \in R[x]$, $t \geq 0, m \geq 1$, $(x-a)^m \mid \left((x-a)^{t+m}q\right)^{(t)}$*

Proof. We prove this lemma by induction.

As a base case, we prove the lemma for $t = 0$.

$$((x-a)^m q)^{(0)} \quad = \quad (x-a)^m q$$

Thus, $(x-a)^m \mid ((x-a)^m q)^{(0)}$

Next, we make the inductive assumption for $t = i$: $(x-a)^m \mid \left((x-a)^{i+m}q\right)^{(i)}$. Using this assumption, we may prove the lemma holds for $t = i+1$.

$$\begin{aligned} \left((x-a)^{m+i+1}q\right)^{(i+1)} &= \left((m+i+1)(x-a)^{m+i}q - (x-a)^{m+i+1}q^{(1)}\right)^{(i)} \\ &= \left((x-a)^{m+i}\left((m+i+1)q - (x-a)q^{(1)}\right)\right)^{(i)} \end{aligned}$$

Thus, by the inductive assumption, $(x-a)^m \mid \left((x-a)^{m+i+1}q\right)^{(i+1)}$. By the inductive principle, our lemma holds. □

Lemma 24. *For all polynomials $q \in R[x]$ such that $(x-a) \nmid q$, $t \geq 0$, $m \geq 1$, $(x-a)^{m-t+1} \nmid ((x-a)^m * q)^{(t)}$*

Proof. We prove this lemma by induction. Note that we may uniquely represent q as $(x-a)b_1 + r$ such that $r \neq 0$ and $\deg(r) < 1$.

As a base case, we prove the lemma for $t = 0$.

$$\begin{aligned} ((x-a)^m q)^{(0)} &= (x-a)^m q \\ &= (x-a)^m((x-a)b_1 + r) \\ &= (x-a)^{m+1}b_1 + (x-a)^m r \end{aligned}$$

Because $(x-a)^m r \neq 0$, $\deg(r) < 1$, and $(x-a)^{m+1} \nmid (x-a)^m r$, we know that $\deg\left((x-a)^m r\right) < \deg\left((x-a)^{m+1}\right)$, and $(x-a)^{m+1} \nmid ((x-a)^m q)^{(0)}$.

Next, we make the inductive assumption for $t = k$: $(x-a)^{m-k+1} \nmid ((x-a)^m * q)^{(k)}$. Using this assumption, we may prove that the lemma holds for $t = k+1$.

$$\begin{aligned} ((x-a)^m * q)^{(k+1)} &= \left(m(x-a)^{m-1}q + (x-a)^m q^{(1)}\right)^{(k)} \\ &= \left((x-a)^{m-1}\left(mq + (x-a)q^{(1)}\right)\right)^{(k)} \end{aligned}$$

Let $m' = m-1$ and $q' = mq + (x-a)q^{(1)}$. We know through the inductive assumption that:

$$\begin{aligned} (x-a)^{m'-k+1} &\nmid \left((x-a)^{m'}q'\right)^{(k)} \\ (x-a)^{(m-1)-k+1} &\nmid \left((x-a)^{m-1}q'\right)^{(k)} \\ (x-a)^{m-(k+1)+1} &\nmid \left((x-a)^{m-1}\left(mq + (x-a)q^{(1)}\right)\right)^{(k)} \\ &\nmid ((x-a)^m * q)^{(k+1)} \end{aligned}$$

Thus, by the inductive principle, our lemma holds. □

Lemma 25. *For all polynomials $q \in R[x]$ such that $(x-a) \nmid q$, $(x-a) \nmid \left((x-a)^t q\right)^{(t)}$.*

Proof. We prove this lemma by induction. Note that we may uniquely represent q as $(x-a)b_1 + r$ such that $r \neq 0$ and $\deg(r) < 1$.

As a base case, we prove the lemma for $t = 0$.

$$\left((x-a)^0 q\right)^{(0)} = q$$

Because $(x-a) \nmid q$, $(x-a) \nmid \left((x-a)^0 q\right)^{(0)}$.

Next, we make the inductive assumption for $t = i$: $(x-a) \nmid \left((x-a)^i q\right)^{(i)}$. Using this assumption, we may prove that the lemma holds for $t = i+1$.

$$\begin{aligned} \left((x-a)^{i+1}q\right)^{(i+1)} &= \left((i+1)(x-a)^i q + (x-a)^{i+1}q^{(1)}\right)^{(i)} \\ &= \left((i+1)(x-a)^i q\right)^{(i)} + \left((x-a)^{i+1}q^{(1)}\right)^{(i)} \end{aligned}$$

By Lemma 23, $(x-a) \mid \left((x-a)^{i+1}q^{(1)}\right)^{(i)}$. Thus, for some unique polynomial $b_2 \in R[x]$, $\left((x-a)^{i+1}q^{(1)}\right)^{(i)} = (x-a)b_2$. By the inductive assumption, $(x-a) \nmid \left((i+1)(x-a)^i q\right)^{(i)}$. Thus, for some unique polynomials $b_3, r_3 \in R[x]$ (such that $r_3 \neq 0, \deg(r_3) < 1$), $\left((i+1)(x-a)^i q\right)^{(i)} = (x-a)b_3 + r_3$.

$$\begin{aligned} \left((x-a)^{i+1}q\right)^{(i+1)} &= ((x-a)b_3 + r_3) + ((x-a)b_2) \\ &= (x-a)(b_3 + b_2) + r_3 \end{aligned}$$

As $r_3 \neq 0, \deg(r_3) < 1$, $(x-a) \nmid \left((x-a)^{i+1}q\right)^{(i+1)}$. By the inductive principle, our lemma holds. □

Lemma 4.

Let $F_j \in R[x]$ $(0 \le j \le d)$ each of degree j such that $\gcd(F_0, \ldots, F_d) = 1$. *For all elements $a \in R$ such that $\forall_{0 \le j \le d}$ $(x-a) \nmid F_j$, $q \in R[X]$ such that $(x-a) \nmid q$, and $r_j \leftarrow R^{m+\deg(q)}[x]$ $(0 \le j \le d)$, and:*

- *if $m > d$, $f = (x-a)^m * q \to (x-a)^{m-d} \mid \sum_{j=0}^{d} f^{(j)} * F_j * r_j \quad \wedge \quad (x-a)^{m-d+1} \nmid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$*
- *if $m \le d$, $f = (x-a)^m * q \to (x-a) \nmid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$*

with overwhelming probability.

Proof.

- If $m \le d$, by Lemma 25, there exists with overwhelming probability at least one index j $(0 \le j \le d)$ such that $(x-a) \nmid ((x-a)^m * q)^{(j)} F_j * r_j$[1]. Let $A = \{j \mid (x-a) \nmid ((x-a)^m * q)^{(j)} F_j * r_j\}$ and $B = \{j \mid 0 \le j \le d \wedge j \notin A\}$. Each polynomial $((x-a)^m * q)^{(j)} F_j * r_j$ can be represented as $(x-a)q_j + s_j$. By the definition of A and B, $\forall_{j \in A}\ s_j \ne 0$ and $\forall_{j \in B}\ s_j = 0$.

$$\begin{aligned}
\sum_{j=0}^{d} ((x-a)^m * q)^{(j)} * F_j * r_j &= \left(\sum_{j \in A} ((x-a)^m * q)^{(j)} * F_j * r_j \right) + \\
&\quad \left(\sum_{j \in B} ((x-a)^m * q)^{(j)} * F_j * r_j \right) \\
&= \sum_{j \in A} ((x-a)q_j + s_j) + \sum_{j \in B} ((x-a)q_j) \\
&= (x-a) \sum_{j=0}^{d} q_j + \sum_{j \in A} s_j
\end{aligned}$$

  Note that $\sum_{j \in A} s_j \ne 0$ with overwhelming probability. Thus, as $\deg\left(\sum_{j \in A} s_j\right) < 1$, we may conclude that $(x-a) \nmid \sum_{j=0}^{d} ((x-a)^m q)^{(j)} * F_j * r_j$ with overwhelming probability.

- If $m > d$, by Lemma 23, $(x-a)^{d-m} \mid f^{(j)}$ for $0 \le j \le d$. Thus, by the distributive property over rings, $(x-a)^{m-d} \mid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$. Note also that by Lemma 24, $(x-a)^{d-m+1} \nmid f^{(d)}$. By the analysis above, with overwhelming probability, $f = (x-a)^m * q \to (x-a)^{m-d+1} \nmid \sum_{j=0}^{d} f^{(j)} * F_j * r_j$.

□

[1] Note that $(x-a) \nmid r_j$ with overwhelming probability, as r_j is random and of polynomial size.

Chapter 5

Hot Item Identification and Publication

Ideally, all privacy-preserving distributed information sharing protocols would use extremely strong definitions of security. In Chapter 4, we constructed protocols for many important applications that are secure under standard notions of cryptographic security. Unfortunately, these protocols are not sufficiently efficient or robust for certain demanding applications; we believe that no protocols so stringently secure can be sufficiently efficient. Thus, in this chapter, we examine variants of the Over-Threshold Set-Union problem from Chapter 4. We construct protocols for these variants, the hot item identification and publication problems, that are both extremely efficient and robust. In order to achieve these levels of efficiency, we compromise privacy. Instead of being cryptographically secure, our protocols achieve the novel notions of owner and data privacy.

We begin in Section 5.1 by describing the hot item identification and publication problems, as well as defining our security properties of owner and data privacy. We then construct efficient and robust protocols for the hot item identification and publication problems in Sections 5.2 and 5.3, respectively. In Section 5.4 we analyze the security of these protocols, before describing several extensions in Section 5.5. To show the utility and efficiency of our hot item identification protocol, we perform experiments in Section 5.6 in which we perform distributed detection of simulated worms in real network traffic.

5.1 Problem Definition and Desired Properties

In this section, we give a formal definition of the hot item identification and publication problems and our novel security and privacy properties.

Problem Definition. In our problem setting, each player i ($1 \leq i \leq n$) in the system holds a private dataset S_i of m elements from a domain denoted M[1]. Let a *k-threshold hot item* (referred to in this chapter as a *hot item*) be any element a that appears in at least k distinct players' private input datasets. H_k is the set of all k-threshold hot items in the system. All items not in H_k are called *cold items*. We define the hot item identification problem as follows: each player i ($1 \leq i \leq n$) learns $S_i \cap H_k$, denoted P_i.

[1] Note that here for simplicity, we assume each player has the same number of items in its private set; our protocols can be easily extended to scenarios where this is not the case.

Adversary Model. In this chapter, we consider a strong adversary model. First, we assume that the adversary can eavesdrop on all communication. Second, we assume that a fraction λ of the players may maliciously and arbitrarily misbehave, while the rest of the players are honest-but-curious.

A malicious player may not follow the protocol; instead he may behave arbitrarily. For example, malicious players could collude such that if λn exceeds k, they can make an arbitrary cold item hot by each reporting it as in their private input set[2]. However, this issue is out of the scope of our problem. Because any player has the freedom to choose his private input set, any protocol in this setting is vulnerable to this manipulation.

Another attack a malicious player could mount is to claim to have many copies of one item to try to boost the frequency of a cold item to be high enough to become a hot item; we call this the *inflation attack*. Thus, our protocols must ensure that during hot item identification, each player can only contribute to the frequency count of an item at most once. Note that this is a challenging task, as we will need to preserve the players' privacy as well as security. We have designed a novel cryptographic method, *one-show tags* (Section 5.2.2), to prevent the inflation attack and preserve the players' privacy. This novel construction could be of independent interest.

Moreover, malicious players could attempt to forge cryptographic signatures, send bogus messages, try to learn honest players' private data, or fool other players. Note that our protocol defends against all of these attacks, by provably achieving the properties of owner and data privacy as defined below.

Definition of Correctness. Given the false positive rate δ_+ and the false negative rate δ_-

- $\forall a \in S_i \cap H_k$, player i learns that $a \in P_i$ with probability at least $1 - \delta_-$,
- $\forall a \in S_i \cap \bar{H}_k$, player i learns that $a \notin P_i$ with probability at least $1 - \delta_+$.

To allow more flexible and efficient protocols, while preserving a reasonable level of privacy for participants, we define two new concepts of security: owner privacy and data privacy.

Definition of Owner Privacy. Intuitively, a protocol that is *owner private* prevents an adversary from determining that any element a is part of a particular honest player's private input set. Even an element that is known to be hot cannot be shown to be held by any player; the elements of a player's private input set are anonymous.

Formally, we say a k-hot item identification protocol satisfies owner privacy if: no coalition of at most λn malicious PPT adversaries can gain more than a negligible advantage in associating an item a with a honest player i ($1 \leq i \leq n$) such that $a \in S_i$, over a situation in which all players simply are given the set of hot items P and their frequencies.

Definition of Data Privacy. *Data privacy* concerns protection of the union of the players' inputs, especially cold items. Ideally, a truly privacy-preserving hot-item identification protocol should not reveal any information about the cold items; we give such a protocol in Chapter 4. However, such a strong privacy definition entails inefficient solutions, as the cryptographic techniques add too much overhead in computation and communication for many situations. Thus, in this chapter, we study the tradeoff between privacy and efficiency; in particular, we rigorously define a relaxed notion of privacy, show that it provides sufficient protection for many applications, and design efficient solutions to achieve it.

[2] In fact, even when λn is less than k, they could make a cold item that appears at least $k - \lambda n$ times in non-malicious players to appear as a hot item.

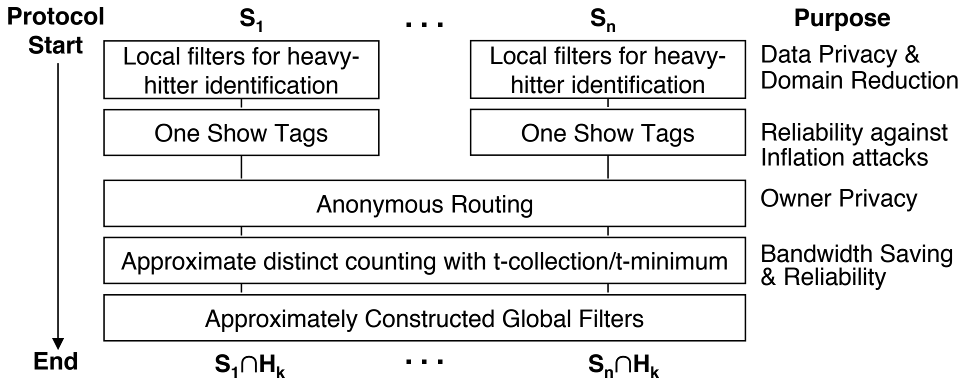

Protocol Start
S_1
. . .
S_n
Purpose
Local filters for heavy-hitter identification
Local filters for heavy-hitter identification
Data Privacy & Domain Reduction
One Show Tags
One Show Tags
Reliability against Inflation attacks
Anonymous Routing
Owner Privacy
Approximate distinct counting with t-collection/t-minimum
Bandwidth Saving & Reliability
Approximately Constructed Global Filters
End
$S_1 \cap H_k$
. . .
$S_n \cap H_k$

Figure 5.1: Components of HotItem-ID protocol: HotItem-ID defines how to efficiently compute an approximate representation of H_k in distributed fashion. Each player i ($1 \leq i \leq n$) constructs local filters to approximately represent his private input set S_i, generates one-show tags for marked bits in filters, and sends a subset of those one-show tags to the network using anonymous routing. A distributed approximate distinct element counting protocol aggregates those tags in the network. At the end of the protocol, all players learn the global filters that approximate H_k. At the right side of the figure we list the purpose of each component.

In our definition of data privacy, the degree of the privacy of an item describes the size of the 'crowd' in which the element is hidden; no adversary can determine which element of the large crowd was included in the players' private input sets. Formally, we say an element a which appears in f_a players' private input sets has $\Phi(f_a)$-degree of data privacy if: no coalition of at most λ malicious probabilistic polynomial-time (PPT) adversaries [28] can distinguish the element a from an indistinguishable set with expected size $\Phi(f_a)$. Thus, for a cold item a, the larger $\Phi(f_a)$ is, the better protected it is in general.

Efficiency. We also want the protocol to be highly efficient. In particular, to identify hot items that appear in at least a certain fraction of the total number of players' private input sets, we would like the protocol to have constant per-player communication overhead. We will show that our protocol achieves this property by using a combination of various approximate counting methods.

5.2 HotItem-ID Protocol

In Figure 5.1, we show an overview of the components of our efficient privacy-preserving hot item identification protocol HotItem-ID and their purposes. We first introduce the intuition behind each component. Then, in Section 5.2.6, we describe the full construction of our HotItem-ID protocol. Once all players learn the hot items in their private datasets, they can run our HotItem-Pub protocol (Section 5.3) to securely publish the identified hot items.

5.2.1 Approximate Heavy-Hitter Detection

A naïve approach to hot item identification is to count the number of players holding each possible element, then determine whether that element is hot. However, performing this task is extremely inefficient for large domains; many applications require use of strings of 1024 bits or more. In most scenarios with a large number of players, it is prohibitively expensive to count each of these 2^{1024} (or more) distinct elements separately. Even if the players only count the frequency of those elements that appear in their private input sets, the bandwidth

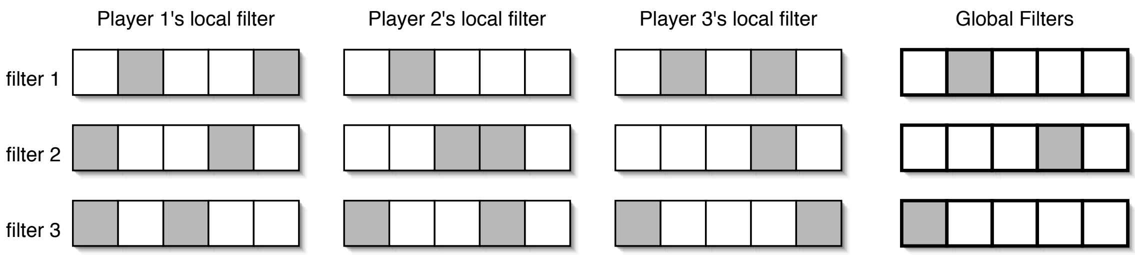


Figure 5.2: In our HotItem-ID protocol, each player i constructs a set of local filters from his private input set S_i (dark bits are 'hit'). The players then construct global filters using an approximate counting scheme; if a bit was hit by at least k players, then it is 'hit' (dark) as well. If an element hashes to a dark bit in each of the global filters, then it is classified as hot.

required will often be prohibitive. In order to avoid this inefficient naïve approach, we utilize an *approximate heavy-hitter identification scheme* [18]. This approximation scheme allows us to efficiently combine the process of counting the elements held by all players.

In this approximate heavy-hitter identification scheme, each player constructs a local filter. The players then combine their local filters to construct a global filter; this global filter approximately identifies hot items. We illustrate this process of local and global filter construction in Figure 5.2.

First, each player constructs a set of T *local filters*, which approximately represent his private input set. Let $h_1, \ldots, h_T : \{0,1\}^\omega \to \{1, \ldots, b\}$ be polynomial-wise independent hash functions with uniformly distributed output, such as cryptographic hash functions [40]. Each filter q ($1 \leq q \leq T$) for player i ($1 \leq i \leq n$) is an array of b bits, represented by the bit array $\{w_{i,q,j}\}_{j \in \{1,\ldots,b\}}$. A local filter bit set to 1 indicates that at least one element in the player's private input set hashes to that bit; we refer to such bits as *hit*. Formally, player i ($1 \leq i \leq n$) computes each bit $w_{i,q,j} := 1 \Leftrightarrow \exists_{a \in S_i} h_q(a) = j$. The players then combine their local filters into a set of *global filters*, using methods described below; global filters approximately represent the players' combined private input sets. We will represent each global filter as the bit array $\{x_{q,j}\}_{j \in \{1,\ldots,b\}}$ ($1 \leq q \leq T$). If at least k players marked bits j of filter q as hit, then let the bit in the global filter $x_{q,j} := 1$ ($1 \leq q \leq T$, $1 \leq j \leq b$). Otherwise, let $x_{q,j} := 0$. Given this global filter, a is hot with high probability if $x_{1,h_1(a)} = 1, \ldots, x_{T,h_T(a)} = 1$. For statistical analysis of this approximation scheme, see Section 5.4.1.

5.2.2 One-Show Tags

In order to construct the global filters, we must count how many players hit each bit in their local filters. Malicious players may attempt to affect this count by 'voting' multiple times for a bit; this *inflation attack* could lead to elements being erroneously marked as hot. If players later publish their hot items, the attacker would learn private information through this attack. Most techniques that ensure players 'vote' at most once would reveal an unacceptable level of information about each players' private filters. The voting process must therefore be anonymous to prevent the adversaries from learning information about any particular players' private input. We ensure that each player can 'vote' only once, without compromising their privacy, with *anonymous one-show tags*. If a player set a bit, he constructs a tag for that bit; the players then count the number of valid tags for each bit to construct the global filters. We require that one-show tags posses the following properties: (a) no PPT adversary can construct more than one valid tag for any bit with non-negligible probability; (b) any PPT player can detect tags that are invalid, or not associated with a particular bit, with overwhelming probabilty; (c) for every bit, the tags constructed by any two players are distinct, with overwhelming probability; (d) no PPT adversary can distinguish the player that constructed it, with probability non-negligibly different than $\frac{1}{n}$, where n is the number of honest players.

Let all players share a group public key for tag verification. Each player holds an individual private key, used to construct tags. The message and one-show parameters vary by bit, to ensure that tags constructed for one bit are not confused with those for another bit. We denote the algorithm for construction of a one-show tag $\texttt{OST}$(group public key, private signing key, message, one-show parameter). We provide a construction of such tags in Appendix B.3. Note that this cryptographic tool is communication-efficient; our construction requires only $1,368$ bits.

5.2.3 Approximate Distinct Element Counting

We may now securely identify hot items by utilizing global filters and anonymous one-show tags. However, exactly counting the number of valid, distinct one-show tags is inefficient; the players would have to collect every single tag. To ensure that no tag is counted twice, the players would also have to store information about every tag.

We can perform the task of approximate counting of distinct tags through an efficient algorithm for *approximate distinct element counting* [4]. In our protocol, however, we gain even more efficiency by estimating directly, through modified use of this approximation, whether there are more or fewer than k tags for a bit; this is the task that must be performed to construct the global filters.

To approximate the number of distinct tags, we need t specific tags from the set. Let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\omega$ be a collision-resistant cryptographic hash function [40]. Let $v_1, \ldots, v_t$ represent valid one-show values that, when hashed with $\mathcal{H}$, have the smallest values of any tags in the set. We estimate the total number of distinct tags as $\frac{t2^\omega}{\mathcal{H}(v_t)}$ [4]. To increase the accuracy of this approximation, one may choose α independent hash functions and perform this estimation with each such function; the resulting estimate is the median of the approximation obtained with each hash function. We have found $\alpha := 1, t := 25$ sufficient in practice.

To perform the task of determining whether there are at least k distinct tags corresponding to a bit, the players can perform a full approximation. However, we have designed a more efficient variant for this particular problem. Note that, if there exist t tags such that the values of their hashes is at most $\frac{t2^\omega}{k}$, the approximation scheme we describe above will conclude that there are at least k total distinct tags. The players thus can instead perform this more efficient collection task, often without even examining ever tag. For example, any tag with value greater than $\frac{t2^\omega}{k}$ may be immediately discarded without affecting the approximation.

In many situations, there is a large gap between the frequency of hot items and cold items. For example, worm attacks will be detected a large number of times by network monitors; normal traffic is far more varied. Thus, to gain greater efficiency in detecting bits with at least k one-show tags, we may adjust the t-collection protocol to follow the algorithm of [54]: each player attempts to collect t tags with hash values of at most $(1+\gamma)\frac{t2^\omega}{k}$. ($\gamma$ is a constant based on the size of the 'gap' between the frequency of hot and cold items.) We gain efficiency in such situations by reducing t, while retaining accuracy. In practice, we have found $\gamma := .2, t := 5$ sufficient. This scheme is identical to the above scheme if $\gamma := 0$.

Protocol: t-Collection
Input: All n players know $\mathcal{H}, t, k, \gamma$, and the diameter of the network, D. Each player i $(1 \leq i \leq n)$ holds a set $U'_{i,0} = U_{i,0}$ of one-show tags for some specific bit.
Output: Each player outputs `success` if they collected t valid one-show tags with one show value v such that $\mathcal{H}(v) \leq (1+\gamma)\frac{t2^\omega}{k}$; otherwise, they output `failure`.

1. For $\ell = 1, \ldots, D$, each player i $(1 \leq i \leq n)$:
   - (a) Sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
   - (b) If $|U_{i,\ell-1}| = t$ then player i **stops participating** in the protocol and goes to Step 2.
   - (c) Receives a set of tags as a result of Step 1a (a) – let the set $U_{i,\ell}$ be those which are valid tags for the correct bit, with one show value v such that $\mathcal{H}(v) \leq (1+\gamma)\frac{t2^\omega}{k}$.
   - (d) Calculates $U'_{i,\ell} = U'_{i,\ell-1} \cup U_{i,\ell}$.
2. Each player i $(1 \leq i \leq n)$ outputs `success` if $|U_{i,\ell}| = t$, and otherwise outputs `failure`.

Figure 5.3: t-collection protocol

5.2.4 Anonymous Communication

We must now apply our tools to a network structure to complete the protocol. In order to disassociate the anonymous one-show tags each player constructs from their location in the network, each player anonymously routes his tags to ρ random players. The constant ρ can be varied to achieve greater or lesser degrees of robustness; at least one participating player should receive each player's tags with high probability. We require an anonymous routing scheme allows any player to send a message to a uniformly randomly selected player, without revealing the source to any intermediate routing node. Simple and lightweight schemes can be used, as we require only that each player send anonymous messages to a uniformly selected destination node, without revealing who sent the message. Some previously proposed anonymous networking schemes include [11, 13, 42, 48, 49].

5.2.5 Distributed One-Show Tag Collection

The players, once they have anonymously received their initial set of tags, then count the number of tags associated with each bit of the filters. There are many different ways in which they can accomplish this task, such as through the use of gossip networks or centralized servers; the ideal method depends on the particular network in which it is employed. We suggest two distributed, secure, and efficient protocols that require only minimal network capabilities: each player must be able to send messages to each of its neighbors, who form a connected graph. As we require only minimal network capabilities, even more efficient schemes than appear in previous work may be employed [47, 50, 53]. For clarity of presentation, our protocols assume synchronous communication, but can be adapted to an asynchronous model by adapting the termination conditions.

Our two protocols for distributed one-show tag collection function as follows: (a) collecting t one-show tags (with sufficiently small hash value) to approximate whether there are at least k valid, distinct tags for a bit or, (b) finding the t one-show tags with the smallest hash values to approximate how many valid, distinct tags there are for a bit. Note that these protocols can be executed in parallel for each bit of the global filters.

We give an efficient, robust protocol for (a), the *t-collection* task in Figure 5.3. Each player maintains a set of at most t valid tags which have hash values at most $(1+\gamma)\frac{t2^\omega}{k}$. Upon learning a new tag that will be added to his set, a player sends the new tag to all of his neighbors. When

Protocol: t-Minimum Aggregation
Input: All n players know $\mathcal{H}, t$, and the diameter of the network D. Each player i ($1 \le i \le n$) holds a set $U'_{i,0} = U_{i,0}$ of one-show tags for some specific bit.
Output: Each player learns the set U', those t tags with the smallest hashes of their one-show values.

1. For $\ell = 1, \ldots, D$, each player i ($1 \le i \le n$):
   (a) Sends the set of new small tags $U_{i,\ell-1} \cap U'_{i,\ell-1}$ to all of his neighbors.
   (b) Receives a set of tags as a result of Step 1a – let those which are valid one-show tags for the correct bit form the set $U_{i,\ell}$.
   (c) Calculates $U'_{i,\ell}$ to be those t tags with one-show values $v_1, \ldots, v_t$ such that $\mathcal{H}(v_1) < \cdots < \mathcal{H}(v_t)$ and $\forall_{w \in (U_{i,\ell} \cup U'_{i,\ell-1}) \cap \overline{U'_{i,\ell}}}\ \mathcal{H}(w) > \mathcal{H}(v_t)$.
2. Each player i ($1 \le i \le n$) outputs the set $U' = U'_{i,D}$.

Figure 5.4: t-minimum value aggregation protocol

a player has collected t tags with hash values of at most $(1+\gamma)\frac{t2^{\omega}}{k}$, and sent each of these tags to his neighbors, he ends his participation in the t-collection protocol. If there do not exist t such small-hash-value tags, the protocol must continue until it converges. The players must ensure that information travels from one side of the network to the other; this requires D rounds, where D is the diameter of the network[3].

We give an efficient, robust protocol for (b), the *t-minimum* task in Figure 5.4. Like the *t-collection* protocol, each player passes on small-valued valid tags to his neighbors, retaining the t tags with the smallest hash values. This process continues until it converges (at D rounds) and all players have learned the t tags with the smallest hash values. This protocol is based on that of [46].

5.2.6 Putting HotItem-ID to Work

We now outline our HotItem-ID protocol in Figure 5.5. Let sk_i be player i's ($1 \le i \le n$) private key, allowing him to construct one-show tags. These keys can be distributed by a trusted 'group manager', by a mutually distrustful group of players acting as such, or by all players acting in concert (see Appendix B.3).

In Step 1, each player i ($1 \le i \le n$) constructs T local heavy-hitter identification filters $\{w_{i,q,j}\}_{j\in\{1,\ldots,b\}}$ ($1 \le q \le T$) from their private input sets. They then construct a one-show tag for each bit in the filters marked as hit ($w_{i,q,j} = 1$); for a specific construction of the one-show value, see Appendix B.3. In Step 2, player i anonymously sends the tags for counting to ρ randomly chosen players. (If the players will be utilizing t-collection for counting, they may save bandwidth by only sending those tags where the hash ($\mathcal{H}$) of the one-show value is at most $(1+\gamma)\frac{t2^{\omega}}{k}$.)

To construct the global filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ ($1 \le q \le T$), the players must determine, for each bit in the global filters, whether at least k players hit that bit in their local filters. In Step 3, to efficiently and securely perform this task, the players utilize either the t-collection protocol (Figure 5.3) or t-minimum aggregation protocol (Figure 5.4). Using one of these protocols, the players approximate whether there were at least k valid, distinct tags constructed for each bit (for greater accuracy, the players perform this approximation α times. taking the median of

[3] Note that this achieves a strict lower bound on the speed of information transmission, based on the definition of network diameter.

Protocol: HotItem-ID

Input: There are n players, λ maliciously colluding, each with a private input set S_i. Each player $i \in \{1, \ldots, n\}$ holds a secret key sk_i allowing him to construct one-show tags, as well as a common public key pk for tag verification. These keys can be chosen by a trusted administrator or by a distrustful collective [3]. $h_1, \ldots, h_T$ are independently chosen cryptographic hash functions with range $\{1, \ldots, b\}$. $\mathcal{H}$ is a cryptographic hash function with range $\{0,1\}^\omega$. $\rho, b, T, t, \alpha, \gamma$ are parameters known to all participants.

Output: Each player i ($1 \leq i \leq n$) obtains the set $P_i \subseteq S_i$, such that each element $a \in P_i$ is a hot item. All players hold the approximate heavy-hitter filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ $(1 \leq q \leq T)$.

1. Each player i ($1 \leq i \leq n$) constructs T local approximate heavy-hitter identification filters $\{w_{i,q,j}\}_{j\in\{1,\ldots,b\}}$ $(1 \leq q \leq T)$ from their private input set S_i: $\exists_{a\in S_i} h_q(a) = j \Rightarrow w_{i,q,j} = 1$. $w_{i,q,j} = 0$, otherwise.
2. Each player i ($1 \leq i \leq n$), for each bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$) s.t. $w_{i,q,j} = 1$:
   (a) constructs a one-show tag $\texttt{OST}(pk, sk_i, q \mid\mid j, \text{one-show-value})$
   (b) player i anonymously routes the tag to ρ randomly chosen players (if the players will be utilizing t-collection in Step 3, do not send the tag unless $\mathcal{H}$ of the one-show value is $\leq (1+\gamma)\frac{t2^\omega}{k}$)
3. For each bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$) , all players $1, \ldots, n$, perform exactly one of the following tasks:
   - perform t-collection, with α independent hash functions, on the tags received in Step 2, attempting to collect, for each hash function $\mathcal{H}$, t valid one-show tags such that each one-show value $v \leq (1+\gamma)\frac{t2^\omega}{k}$. If t-collection was successfully performed for the majority of hash functions, set $x_{q,j} = 1$. Otherwise, $x_{q,j} = 0$.
   - perform t-minimum value aggregation, with α independent hash functions, on the tags received in Step 2. If v is the tth minimum hash value, approximate that there are $\frac{t2^\omega}{v}$ total tags; the median of each such approximation forms the final approximation. If this process determines that there were at least k tags, set $x_{q,j} = 1$. Otherwise, $x_{q,j} = 0$.
4. Each player calculates his output set P_i: for each element $a \in S_i$, if $\forall_{q\in\{1,\ldots,T\}}\ x_{q,h_q(a)} = 1$, then $a \in P_i$.

Figure 5.5: HotItem-ID protocol, for identifying the hot items in each players' private input.

the approximations). If the players conclude that there were at least k valid, distinct tags constructed for bit j of filter q ($1 \leq j \leq b$, $1 \leq q \leq T$), then they set $x_{q,j} := 1$; else, set $x_{q,j} := 0$.

Once they have constructed the global filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ $(1 \leq q \leq T)$, each player can utilize these filters, as shown in Step 4, to determine whether each element of his input is a hot item. If, for an element a, $\forall_{q\in\{1,\ldots T\}}\ x_{q,\mathcal{H}(a)} = 1$, then a is a hot item with high probability.

5.3 Hot Item Publication Protocol

In this section, we present our HotItem-Pub protocol. This protocol utilizes the HotItem-ID protocol, so that each player may identify his hot items, but also ensures that the players may securely and efficiently publish their common hot items.

5.3.1 Commitment to Foil Attacks

When players publish hot items, we must prevent two attacks by malicious players: (1) *suppressing* hot items, so players do not learn them; (2) *fooling* honest players into accepting cold items as hot. As we have shown, our HotItem-ID protocol (see Section 5.2) effectively allows all players to identify the hot items in their private input sets, even in the presence of malicious players. By using the HotItem-ID protocol to identify hot items, we must only address attacks on the publication process itself.

Common Protocol Outline. Each of the variants of our HotItem-Pub protocol prevents both item suppression and fooling attacks; however, they achieve different levels of owner privacy. For clarity, we will first describe the basic outline for the protocol, which is common between all variants. In the HotItem-Pub protocol, the players follow four steps:

1. *Commitment.* Each player constructs a computationally-binding and computationally-hiding *commitment* to their private input set [40]; the exact form of the commitment varies according to the level of owner-privacy desired. This is later used to prove (by *opening* the commitment) that a published hot item was held by some player before the global filters were constructed, without revealing any information at this stage of the protocol. If owner-privacy is desired, the players then send each of these commitments to ρ random players. This commitment is then distributed to all players.
2. HotItem-ID. The players then execute the HotItem-ID protocol, described in Section 5.2. Each player learns, with high probability, which of the elements of his private input set are hot, even in the presence of malicious players. In addition, each player also obtains the global filters $\{x_{q,j}\}_{j\in\{1,\ldots,b\}}$ $(1 \le q \le T)$ used to identify their hot items.
3. *Publication.* Each player, for each hot item in his input, constructs an opening for the commitment to that element (see the Commitment step). If owner-privacy is desired, the players then send each of these element/opened commitment pairs to ρ random players. All players the use a *redundant element distribution* protocol, ElementDist, to efficiently and robustly publish the element/opened commitment pairs. Such a protocol follows from the following rule: when a player receives a previously-unseen hot item, he: (1) using the opened commitment attached to the hot item, checks that he received a valid commitment to that item during the commitment phase of the protocol; (2) if it passes the check, sends the hot item (and associated commitment opening) to each of his neighbors and retains it himself in the set P. By following this simple protocol, duplicate publications of hot items are efficiently suppressed, while ensuring that all players receive each hot item in their *provisional set* P.
4. *Verification.* As a result of publication, each player obtains a provisional set of elements P, each of which may be hot. For each such item $a \in P$, the player checks that it passes the global hot item identification filters; if $\forall_{q\in\{1,\ldots,T\}}\ x_{q,h_q(a)} = 1$, then it is truly a hot item.

5.3.2 Putting HotItem-Pub to Work

We now describe the details of each variant of our HotItem-ID protocol (with varying degrees of owner privacy), as well as the commitments and openings used to meet each definition. Note that to increase the level of owner privacy provided, the players must utilize more bandwidth. We formally describe the HotItem-Pub protocol in Figure 5.6.

No Owner Privacy. If the players are not concerned about owner privacy, they may simply commit to their private input sets using Merkle hash trees. A Merkle hash tree is a data structure allowing a player to produce a constant-sized commitment to an ordered set S. Later any player holding S may produce a commitment opening: a verifiable proof (given the committment) that an element $a \in S$ [41].

Correlated Owner Privacy. In correlated owner privacy, the players wish to ensure that the elements of their private input sets cannot be traced to them, but do not care if an adversary can determine that some pair of elements came from the same player. To achieve this level of privacy, the players may also use Merkle hash tree commitments [41]. The protocol only differs from the non-owner private version in that the players anonymously send their commitments and hot items to random players before they are distributed in the commitment and publication phases of the protocol. This ensures that the elements can not be associated with any player.

Uncorrelated Owner Privacy. In uncorrelated owner privacy, the players wish to ensure that the elements of their private input sets cannot be traced to them, and *also* care if an adversary can determine that some pair of elements came from the same player. When the players desire this level of privacy, they cannot utilize the efficient Merkle commmitments, but instead must commit to each element separately. As the commitments, as well as the hot items, are sent independently and anonymously to random players before publication, they cannot be linked with the player who constructed them. In addition, as elements have independent commitments, no player can gain advantage in determining whether one player held both elements.

5.4 Analysis

Now we proceed to analyze the security and performance of our HotItem-ID and HotItem-Pub protocols.

5.4.1 HotItem-ID Correctness

In Theorem 26 we prove that our HotItem-ID protocol functions correctly: hot items are identified as hot and cold items are identified as cold with high probability. Note that the filter sizes b, T must be chosen as described in the theorem for our guarantees to hold, though they may be smaller in practice.

Theorem 26. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{m \lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

We defer the proof of Theorem 26 to Appendix B.2.1.

5.4.2 Privacy in HotItem-ID

In these theorems, we prove the owner and data privacy of our HotItem-ID protocol.

Theorem 27. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player*

Protocol: HotItem-Pub

Input: There are n players, λ maliciously colluding, each with a private input set S_i. Each player $i \in [n]$ holds a private key sk_i, allowing him to construct one-show tags. $h_1, \dot{,} h_T$ are independently chosen cryptographic hash functions. ρ, b, T, t are parameters known to all participants.
Output: The set P of hot items published by honest players.

1. Each player i $(1 \leq i \leq n)$ commits to their private input set S_i:
    - If the players are not concerned with Owner-Privacy, each player constructs a hash tree from his randomly-permuted private input set S_i, with root hash-value y_i; then sends y_i to all players
    - If the players wish to ensure *Correlated Owner-Privacy*, each player constructs a hash tree from his randomly-permuted private input set S_i, with root hash-value y_i; then anonymously routes y_i to all players
    - If the players wish to ensure *Uncorrelated Owner-Privacy*, for each element $a \in S_i$, he constructs a commitment to a, and anonymously routes it to every other player
2. All players $1, \ldots, n$ execute the HotItem-ID protocol (Figure 5.5), so that each player i $(1 \leq i \leq n)$ obtains: (a) the set $P_i \subseteq S_i$ of hot items in that player's private input set and (b) global hot item identification filters $\{x_{q,j}\}_{j \in \{1,\ldots,b\}\{1,\ldots,b\}}$ $(1 \leq q \leq T)$
3. All players $1, \ldots, n$ publish the hot items in their private input sets:
    - If the players are not concerned with Owner-Privacy:
        (a) for each element $a \in P_i$, constructs a proof of inclusion in S_i, showing the path from the leaf a to the root value y_i
        (b) distributes the elements of the set P_i to all other players, along with their proofs of correctness, using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
    - If the players wish to ensure *Correlated Owner-Privacy*:
        (a) for each element $a \in P_i$, constructs a proof of inclusion in S_i, showing the path from the leaf a to the root value y_i and anonymously routes it to ρ randomly chosen players.
        (b) the players distribute all the elements and proofs received in Step 3a using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
    - If the players wish to ensure *Uncorrelated Owner-Privacy*:
        (a) for each element $a \in P_i$, opens his commitment to a, and anonymously routes it to ρ randomly chosen players.
        (b) the players distribute all the elements and proofs received in Step 3a using the ElementDist protocol, such that all connected honest players learn the set $P = P_1 \cup \cdots \cup P_n$.
4. Each player $i = 1, \ldots, n$ verifies that each element $a \in P$:
    - was committed to with a valid committment by some player in Step 1
    - passes the filter – $\forall_{q \in [T]}\ x_{q,h_q(a)} = 1$

Figure 5.6: HotItem-Pub protocol, for publishing the hot items in each players' private input.

sent any given anonymous message with probability more than negligibly different from a random guess. In the HotItem-ID *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i $(1 \leq i \leq n)$.*

We defer the proof of Theorem 27 to Appendix B.2.2.

When considering data privacy, we wish to prove that the cold elements in $S_1 \cup \cdots \cup S_n$ remain hidden from adversaries. For example, in the HotItem-ID protocol, no player or coalition of at most λ malicious players may gain more than an expected $I = \frac{tnm}{k} \lg b$ bits of information about all players' inputs $\bigcup_{i=1}^{n} S_i$. We prove a tighter bound on the degree to which each element is hidden in a crowd of *indistinguishable elements*. Two elements are indistinguishable if an attacker cannot distinguish which one was in players' private input sets based on the information gained in HotItem-ID. For an element a, its indistinguishable set consists of all elements indistinguishable from it by an adversary who has no prior knowledge. To provide data privacy, we wish for cold items to have large indistinguishable sets; we compromise perfect (semantic) security, in which all items are indistinguishable, to achieve much greater efficiency.

Theorem 28. *In the* HotItem-ID *protocol, each element a, which appears in f_a distinct players' private input sets, has an indistinguishable set of expected size* $\Phi(f_a) = \sum_{\ell=1}^{T} \binom{T}{\ell} \left(1 - \frac{t}{k}\right)^{f_a(T-\ell)} \left(1 - \left(1 - \frac{t}{k}\right)^{f_a}\right)^{\ell} \frac{|M|}{b^{\ell}}$.

We defer the proof of Theorem 28 to Appendix B.2.3.

We graph the fraction of the indistinguishable set size to the domain size, $\frac{\Phi(x)}{|M|}$ in Figure 5.7. When an item a appears in f_a players' private datasets, the higher $\frac{\Phi(f_a)}{|M|}$ indicates that it is harder for adversaries to distinguish a from other items in the domain $|M|$. Note that if a appears in only a few players' private input sets, a very large proportion of the domain is indistinguishable from a. As f_a approaches $\frac{k}{t}$, the size of the indistinguishable set decreases; this character ensures that truly rare elements are highly protected. In many applications, there is a big gap between the frequency of hot items and cold items. In this case, our protocol guarantees that the cold items will be extremely well protected, as their fequency will be much smaller than $\frac{k}{t}$.

5.4.3 Privacy in HotItem-Pub

We now consider the degree of owner privacy conferred by each version of our HotItem-Pub protocol. Correlated owner privacy allows adversaries to link the items published by each player, while uncorrelated owner privacy prevents this.

Theorem 29. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Correlated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Theorem 30. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Uncorrelated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$ for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Additionally, given two elements $a, a' \in P$, no coalition of at most λ malicious players can gain more than a negligable advantage in determining if there exists a honest player i ($1 \leq i \leq n$) such that $a, a' \in S_i$, assuming that the adversary is given the set of hot items P, and the frequency of each hot item.

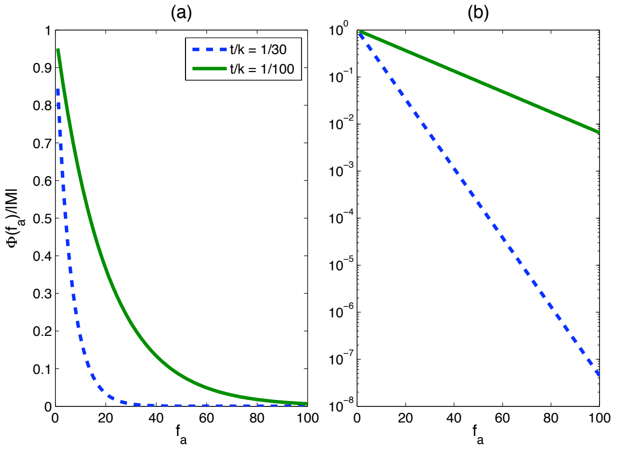


Figure 5.7: The degree of data privacy for an element with frequency f_a is $\frac{\Phi(f_a)}{|M|}$. Ideally, the degree of data privacy would be 1 for all frequencies, but by compromising this strong definition of security, we obtain more efficient and robust protocols. We graph the degree of data privacy **(a)**, showing the increase in protection for rare elements. The same function is graphed in **(b)** on a logarithmic scale, for increased detail.

We prove these theorems in Appendix B.2.3.

5.4.4 Performance

By utilizing a novel combination of approximation counting algorithms, our protocol represents a clear gain in efficiency over exact counting. In particular, aside from the cost of anonymous routing messages, which vary according to the scheme employed, our protocol achieves a multiplicitive factor of $\frac{Tt}{k}$ more efficiency than the exact-counting based non-privacy-preserving protocol using the same message propagation abilities. (A network with improved capability for routing messages will increase the performance of both protocols.) Note that this improvement in performance is significant, especially considering that the baseline protocol provides no privacy at all, where our protocols enforce principled guarantees of privacy while retaining efficiency. In particular, if k is a fraction of n, our protocol achieves constant per player communication overhead. We present experimental results to validate the efficiency of our protocols in distributed networks in Section 5.6.

5.5 Extensions

In this section, we briefly describe several extensions to our HotItem-ID and HotItem-Pub protocols.

Private Input Multisets. Instead of identifying hot items as those that appear in at least k players' private input sets, we may identify those items that appear at least k times in the players' private input multisets. In the modified protocol, we associate γ one-show values with a single (filter,bucket) pair so that up to γ hits to a single bucket by a single player can be counted as distinct elements. The constants b, T may be chosen by slightly adjusting the analysis given in the proof of Theorem 26.

Using Bloom Filters. Bloom filters provide a compact probabilistic representation of set membership [6]. Instead of using T filters, we can use a combined Bloom filter. This achieves the same asymptotic communication complexity, however, in practice the Bloom filter approach can have smaller constants. We describe our approach using T filters in this paper in the interests of clarity. Our scheme can be easily adapted to use Bloom filters. The choices of the constants b, T can be adjusted to give an acceptable false-positive rate, given the domain M; the analysis may be performed similarly to that of [6] and the analysis of the HotItem-ID protocol given in this paper.

Theorem 31. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{mT \lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T} \right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

We defer the proof of Theorem 31 is given in Appendix B.2.4. As the properties of owner and data privacy are unchanged by this variant, we do not perform a separate analysis.

Top-k. The Top-k problem is identifying those k elements that appear most often in the players' private input sets. An efficient non-private solution to this problem is given in [10]. In their protocol, a trusted authority calculates an estimated threshold value τ; every element in the top k appears at least τ times.

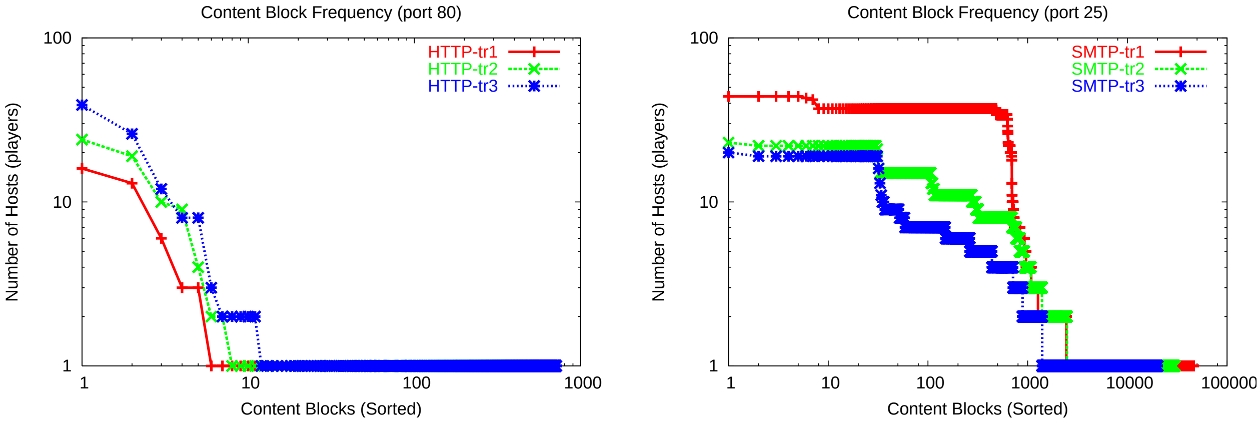
We now briefly describe how to calculate the top k values without use of a central authority and while preserving each players privately. Like in HotItem-ID, the players calculate global filters, but by using t-minimum value aggregation instead of t-collection. In this way, each player may estimate the number of players who hit each (filter,bucket) pair. Using these estimates, the players may approximate τ, marking each (filter,bucket) pair hit by at least τ players as 1. The global filters may then be used to identify the top-k.

5.6 Experimental Results

In order to experimentally evaluate the efficiency and accuracy of our protocol, we implemented our HotItem-ID protocol with t-collection and applied it to distributed worm signature detection. As we describe in Section 5.6.3, our experimental results support the efficiency and accuracy of our protocol.

5.6.1 Distributed Worm Signature Detection

A widely utilized tactic for detecting worms is to search for byte strings that appear with unusually high frequency in network traffic [35, 52]. By distributing the execution of this strategy over a large number of hosts, players can increase the accuracy of their results [35].



(1) HTTP (port 80) **(2) SMTP (port 25)**

Figure 5.8: Number of hosts (players) that have generated each content block (item) from observed suspicious flows.

However, as network traffic often contains sensitive information such as email or information that could aid a network attack, it is imperative to protect the privacy of participants.

In a distributed network monitoring scenario, each monitor attempts to detect malicious traffic on his own network. As anomaly detection is imperfect, this process will often identify innocuous traffic as malicious. Thus, monitors must take further steps to improve the accuracy of their results; comparing the results with other monitors. If the malicious traffic belongs to a worm, there will be other monitors that observed similar traffic; an effective heuristic for identifying possibly malicious traffic is that worms often send a large volume of repetitive data as they attempt to infect other computers. We use our HotItem-ID protocol to perform this comparison while protecting the privacy of non-worm network traffic.

Because polymorphic worms often change their form, players must compare many small pieces of a possible worm payload instead of the entire payload. We use a content-based payload partitioning technique (proposed in [35]) to split a payload into many small segments. In this section, we call such segments *content blocks*. The payload partitioning technique is robust against small payload byte changes and generates the same content blocks for different forms of a worm. Once monitors have identified possible worms and split them into content blocks, they then perform hot item identification over the sets of content blocks. Content blocks generated from innocuous, private network traffic appear only in the sets from a few monitors. Hot content blocks indicate that the traffic has been seen at an unusually large number of hosts; it is therefore almost certainly part of a worm attacking those hosts and can be used as a signature of the worm.

5.6.2 Real-world Data and Experiment Method

We performed simulated distributed network monitoring on traces captured on a campus network that uses one third of a class B IP address space. `HTTP-tr1`, `HTTP-tr2`, and `HTTP-tr3` are one-hour long traces containing all HTTP (tcp port 80) packets and payloads addressed to hosts in the monitored network. Similarly, `SMTP-tr1`, `SMTP-tr2`, and `SMTP-tr3` contain SMTP (tcp port 25) packets captured for an one-hour time period. During the trace collection periods, a total of 5246 IP addresses received at least one packet. Our trace does not contain any known

		`HTTP-tr3`						`SMTP-tr1`					
		# of messages			bandwidth			# of messages			bandwidth		
		t=3	t=5	t=10	t=3	t=5	t=10	t=3	t=5	t=10	t=3	t=5	t=10
ψ=5	naïve	1 (11,399msgs)			1 (228KB)			1 (300,757msgs)			1 (6015KB)		
	γ=0	$\underline{0.044}$	0.075	0.151	$\underline{0.412}$	0.708	1.422	$\underline{0.026}$	$\underline{0.039}$	0.073	$\underline{0.248}$	$\underline{0.364}$	0.687
	γ=0.2	**0.048**	0.081	0.162	**0.455**	0.764	1.524	$\underline{0.032}$	$\underline{0.048}$	0.088	$\underline{0.306}$	$\underline{0.448}$	0.825
	γ=0.5	0.052	0.088	0.177	0.493	0.831	1.664	$\underline{0.042}$	**0.062**	0.117	$\underline{0.395}$	**0.583**	1.100
ψ=10	naïve	1 (25,643msgs)			1 (513KB)			1 (676,612msgs)			1 (13,532KB)		
	γ=0	$\underline{0.038}$	0.068	0.141	$\underline{0.363}$	0.638	1.331	$\underline{0.017}$	$\underline{0.021}$	0.065	$\underline{0.162}$	$\underline{0.199}$	0.611
	γ=0.2	**0.042**	0.071	0.146	**0.394**	0.665	1.372	$\underline{0.022}$	$\underline{0.028}$	0.068	$\underline{0.208}$	$\underline{0.261}$	0.638
	γ=0.5	0.044	0.074	0.149	0.414	0.695	1.402	$\underline{0.036}$	**0.062**	0.124	$\underline{0.341}$	**0.584**	1.169

Figure 5.9: Normalized bandwidth consumption per player in performing hot item identification ($k = 100$). Underlines values indicate that there were false positives or false negatives.

worm traffic and does not contain hot items (threshold $k = 100$). We graph the number of hosts who generate each innocuous content block (by identification of possible worm traffic) in Figure 5.8. Only at most 2.2% (HTTP) or 1.4% (SMTP) of content blocks appear at more than one host. Through manual examination, we determined that content blocks that appeared at more than one host indicated web crawler (HTTP) or spamming (SMTP) activity, and never appeared at more than 45 hosts.

We injected simulated worm traffic into the input of 200 of 1024 monitors in the network. Each monitor generates a set of content blocks from captured anomalous traffic. Note that all the sets generated from those 200 attacked monitors include the content blocks from the worm traffic. We ran our HotItem-ID protocol in the overlay network of 1024 monitors and measured the number of messages and the required bandwidth per monitor, while varying the t-collection parameters t and γ, and the average number of neighbors ψ in the overlay network. We also computed the false positive and false negative rates at the end of the HotItem-ID execution by counting the innocuous content blocks identifed as to be hot (false positives), and the worm content blocks that are not identified (false negatives). In our experiment, we utilized the parameters $b := 606, T := 5$ for HTTP traces, and $b := 4545, T := 5$ for SMTP traces, chosen according to the guidelines in Section 5.4.1. We compared our protocol to a non-private naïve protocol, in which all content blocks are forwarded to all participating 1024 monitors, using the same network topology and the same communication model for both the naïve protocol and the HotItem-ID protocol.

5.6.3 Bandwidth Consumption and Accuracy

Our HotItem-ID protocol efficiently identified every simulated worm injected into the network traces, while generating no false positives; no innocuous data was mistakenly categorized as malicious except when we use `SMTP-tr1` with $t \leq 3$ and $\gamma \geq 0.5$. We present our comparison of the required bandwidth and messages in Figure 5.9. `HTTP-tr3` contains 724 unique content blocks, while `SMTP-tr1` contains 46,120 unique ones. Underlined values in the figure indicate there were false negatives (failed to identify worm content blocks). Our HotItem-ID protocol scales better than the naïve protocol (Section 5.2); as problems increase in size, our protocol becomes more attractive. Even at small problem sizes, the overhead required to protect the privacy of participants is not high. Our experiments show that to ensure correctness while retaining efficiency, we may set $\gamma := 0.2, t := 3$ (HTTP) and $\gamma := 0.5, t := 5$ (SMTP). Our HotItem-ID implementation, based on an efficient group signature scheme [7], requires 193 bytes per message (Appendix B.3) while the naïve protocol utilizes only 20 bytes per message. [4] However, our protocol requires only a small number of message transmissions; as a result, HotItem-ID used only 39% and 58% of the bandwidth used by the naïve protocol in the `HTTP-tr3` and `SMTP-tr1` experiments, respectively. Note that the performance gain from our HotItem-ID protocol increases as the more monitors participate. For example, when 10240 monitors participate in worm signature generation and we need only 10% of them to catch worm traffic ($k = 1000$), our HotItem-ID protocol uses less than 6% of the bandwidth used by naïve protocol.

[4] To save bandwidth and give a trivial measure of privacy against casual attacks [37], we hash each content block with SHA-1 in the naïve protocol.

Chapter 6

Conclusion

In this thesis, we have introduced multiple techniques and protocols for privacy-preserving distributed information sharing. We designed composable set and multiset operations, and protocols for Set-Intersection, Over-Threshold Set-Union, Cardinality Set-Intersection, Threshold Set-Union, Subset, and CNF formula evaluation based on these operations. We then examined the problems of hot item identification and publication, variants of the Over-Threshold Set-Union problem. In order to increase the efficiency and robustness of our hot item protocols, we designed these protocols to achieve novel definitions of security and privacy.

Bibliography

[1] Rakesh Agrawal, Alexandre Evfimievski, and Ramakrishnan Srikant. Information sharing across private databases. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 86–97, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-634-X. 1.1, 2.1

[2] M. Ajtai, J. Komlos, and E. Szemeredi. An $o(n \log n)$ sorting network. In *Proc. of STOC*, pages 1–9, 1983. 1.1

[3] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270, 2000. URL `citeseer.ist.psu.edu/ateniese00practical.html`. 5.2.5

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 1–10, London, UK, 2002. Springer-Verlag. ISBN 3-540-44147-6. 2.2, 5.2.3, B.2.1, B.2.1, 34, B.2.1

[5] M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of STOC*, 1988. 2.1, 4.5.3

[6] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970. 5.5

[7] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *Proceedings of Computer and Communications Security (CCS)*, pages 168–177, 2004. 5.6.3, B.3

[8] Fabirce Boudot, Berry Schoenmakers, and Jacques Traore. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111:77–85, 2001. 2.1

[9] Jan Camenisch. Proof systems for general statements about discrete logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, Mar 1997. 3.2.1, 4.4.1

[10] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 206–215, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-802-4. 1.2, 5.5

[11] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptol.*, 1(1):65–75, 1988. ISSN 0933-2790. 5.2.4

[12] D. Chaum, J.-H. Evertse, J. Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *Advances in Cryptology—CRYPTO '86*, pages 200– 212. Springer-Verlag, 1987. B.3

[13] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–8, 1981. 3.2.2, 4.2.2, 4.3.1, 5.2.4

[14] David Chaum, Jan-Hendrick Evertse, Jeroen van de Graaf, and Rene Peralta. Demonstrating possession of a discrete log without revealing it. In A.M. Odlyzko, editor, *Proc. of Crypto*, pages 200–212. Springer-Verlag, 1986. 3.2.1, 4.4.1

[15] R. Cramer, I. Damgård, and J. Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of Eurocrypt*, pages 280–99. Springer-Verlag, 2001. 3.2.1, 4.4.1

[16] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret sharing scheme. In *Proc. of Eurocrypt*. Springer-Verlag, May 2000. 2.1

[17] Y. Desmedt and K. Kurosawa. How to break a practical mix and design a new one. In *Proc. of Eurocrypt*, pages 557–72. Springer-Verlag, 2000. 3.2.2, 4.2.2, 4.3.1

[18] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, August 2003. 5.2.1

[19] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39:77–85, 1996. 2.1

[20] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag. ISBN 0-387-18047-8. B.3

[21] P. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting of lotteries. In *Proc. of Financial Cryptography*, 2000. 1.2, 3.2.1

[22] Pierre-Alain Fouque and David Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *Proc. of Asiacrypt*, pages 573–84, 2000. 1.2, 3.2.1

[23] Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *Proc. of Eurocrypt*, volume LNCS 3027, pages 1–19. Springer-Verlag, May 2004. 1.1, 2.1, 4, 4.1.1, 4.2.2

[24] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *Proc. of Crypto*, pages 368–87. Springer-Verlag, 2001. 3.2.2, 4.2.2, 4.3.1

[25] Archana Ganapathi and David Patterson. Crash data collection: A windows case study. In *To Appear in the Proceedings of the International Conference on Dependable Systems and Networks*, June 2005. 1.2

[26] Rosario Gennaro and Victor Shoup. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15:75–96, 2002. 3.2.1

[27] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 331–342, June 1998. 2.2

[28] Oded Goldreich. *The Foundations of Cryptography – Volume 2*, volume 2. Cambridge University Press, May 2004. URL `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`. 1.2, 2.1, 3.1, 3.1.2, 4.5.3, 5.1

[29] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–99, 1984. 3.2.1

[30] Qiang Huang, Helen Wang, and Nikita Borisov. Privacy-preserving friends troubleshooting network. In *Proceedings of the Symposium on Network and Distributed Systems Security*, February 2005. 1.2, 2.2

[31] M. Jakobsson. A practical mix. In *Proc. of Eurocrypt*, pages 448–61. Springer-Verlag, 1998. 3.2.2, 4.2.2, 4.3.1

[32] R. Karp, S. Shenker, and C. Papadimitriou. A simple slgorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28(1):51–55, 2003. 2.2

[33] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *Proc. of Crypto*. Springer-Verlag, 2004. 4.4.1

[34] Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In *Proc. of Financial Cryptography*, 2005. 2.1

[35] Hyang-Ah Kim and Brad Karp. Autograph: Toward Automated, Distributed Worm Signature Generation. In *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, San Diego, CA, August 2004. 5.6.1

[36] Lea Kissner, Alina Oprea, Michael Reiter, Dawn Song, and Ke Yang. Private keyword-based push and pull with applications to anonymous communication. In *Applied Cryptography and Network Security*, 2004. 4.3.2

[37] Patrick Lincoln, Phillip Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proceedings of the 13th USENIX Security Symposium*, page 239254, August 2004. 1.2, 2.2, 4

[38] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Proc. of Asiacrypt*, pages 416–33, 2003. 2.1

[39] Philip MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *Proc. of Eurocrypt*, pages 382–400. Springer-Verlag, 2004. 4.4.1

[40] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996. 4.1, 5.2.1, 5.2.3, 1, B.2.1, B.3

[41] Ralph C. Merkle. A certified digital signature. In *Proc. of Advances in Cryptology*, pages 218–238. Springer-Verlag New York, Inc., 1989. ISBN 0-387-97317-6. 5.3.2, 5.3.2

[42] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. Ap3: Cooperative, decentralized anonymous communication. In *11th ACM SIGOPS European Workshop*, September 2004. 5.2.4, B.2.1

[43] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. ACM Symposium on Theory of Computing*, pages 245–54, 1999. 2.1

[44] A. Neff. A verifiable secret shuffle and its application to e-voting. In *ACM CCS*, pages 116–25, 2001. 3.2.2, 4.2.2, 4.3.1

[45] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proc. of Asiacrypt*, pages 573–84, 2000. 1.2, 3.2.1

[46] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: secure information aggregation in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM Press, 2003. ISBN 1-58113-707-9. 5.2.5, B.2.1

[47] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, University of California at Berkeley, Berkeley, CA, 2000. URL `citeseer.ist.psu.edu/ratnasamy01scalable.html`. 5.2.5

[48] M G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communication, Special Issue on Copyright and Privacy Protection*, 1998. 5.2.4

[49] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998. URL `citeseer.ist.psu.edu/article/reiter97crowds.html`. 5.2.4

[50] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001. 5.2.5

[51] Victor Shoup. *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, 2005. URL `http://shoup.net/ntb/`. 4.6.1

[52] Sumeet Singh, Cristian Estan, George Varghese, and Stegan Savage. Automated worm fingerprinting. In *Proceedings of 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, San Francisco, CA USA, December 2004. 5.6.1

[53] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. on Networking*, 11:17–32, February 2003. 5.2.5

[54] Shobha Venkataraman, Dawn Song, Phil Gibbons, and Avrim Blum. New streaming algorithms for superspreader detection. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, February 2005. 5.2.3

[55] Helen J. Wang, Yih-Chun Hu, Chun Yuan, Zheng Zhang, and Yi-Min Wang. Friends troubleshooting network: Towards privacy-preserving, automatic troubleshooting. In *Proceedings of IPTPS*, February 2004. 1.2, 2.2

[56] Andrew C-C Yao. Protocols for secure computations. In *Proc. of FOCS*, 1982. 2.1

Appendix A

Appendices for Privacy-Preserving Set Operations

A.1 Notation

- P – the set of elements which can be members of a private input set
- k – size of each private input set
- n – number of players participating in a protocol
- t – threshold number, an element must appear t times in the private input sets to be included in the threshold set
- $E_{pk}(\cdot)$ – encryption under the additively homomorphic, public key cryptosystem to which all players share a secret key
- $E_{pk}(a) +_h E_{pk}(b)$ – combination of two ciphertexts (under the homomorphic cryptosystem) to produce a re-randomized ciphertext which is the encryption of $a + b$
- $a \times_h E_{pk}(b)$ – combination of an integer and a ciphertext (under the homomorphic cryptosystem) to produce a re-randomized ciphertext which is the encryption of ab
- $f *_h E_{pk}(g)$ – combination of two polynomials (under the homomorphic cryptosystem) to produce a re-randomized encrypted polynomial which is the encryption of $f * g$
- $F_0, \ldots, F_d$ – public 'helper' polynomials for computing element reduction
- $h(\cdot)$ – a cryptographic hash function from $\{0,1\}^*$ to $\{0,1\}^\ell$ ($\ell = \lg\left(\frac{1}{\epsilon}\right)$), where ϵ is negligible.
- $\mathrm{Rd}_d(S)$ denotes the element reduction by d of set S
- $R^a[x]$ denotes the set of all polynomials of degree between 0 and a with coefficients from R
- $[c]$ for an integer c denotes the set $\{1, \ldots, c\}$
- $a := b$ denotes that the variable a is given the value b
- $a \mid\mid b$ denotes a concatenated with b
- $a \leftarrow S$ denotes that element a is sampled uniformly from set S
- $f * g$ is the product of the polynomials f, g
- $\deg(p)$ is degree of polynomial p
- $p^{(d)}$ is the dth formal derivative of p
- $\gcd(p, q)$ is the greatest common divisor of p, q
- S_i is the ith player's private input set
- V_j is the jth element of the set V, under some arbitrary ordering

Appendix B

Appendices for Hot-Item Identification

B.1 Notation

Notation.

- $a \mid\mid a'$ - a and a' concatenated
- $a := a'$ - the value of a' is assigned to a

Variable List.

- a - an item, member of a private input set
- b - number of buckets per filter
- d - $d(x) = \frac{\Phi(x)}{|M|}$, a measure of data privacy which varies by the number of players holding any particular item
- e_i - group certificate for player i $(1 \leq i \leq n)$ in group signature for one-show tags scheme
- f_a - number of players' private input sets in which an element a appears
- g_1, g_2 - group elements in one-show tag scheme
- $g_{q,j}$ - group element for producing one-show value $(1 \leq q \leq T,\ 1 \leq j \leq b)$
- $\mathcal{H}(\cdot)$ - a cryptographic hash function with range $\{0,1\}^\kappa$
- $h_1(\cdot), \ldots, h_T(\cdot)$ - independent cryptographic hash functions with range $[b]$
- i - index over players $(1 \leq i \leq n)$
- I - expected number of bits of information that any coalition can learn about players' private input sets
- j - index over buckets $(1 \leq j \leq b)$
- k - minimum number of players that must hold an alert for it to be published
- ℓ - index
- m - maximum size of S_i
- M - domain of elements $(\bigcup_{i=1}^{n} S_i \subseteq M)$
- n - the number of players
- o - a secret value used in constructing one-show tags
- p_e - probability of edge existence in a random network graph
- p_t - probability that an anonymously routed message is traced to its source
- P_i - each player's set of alerts for publication

- $P = \bigcup_{i=1}^{n} P_i$
- q - index over filters ($1 \leq q \leq T$)
- r_1, r_2, r_1', r_2' - prime numbers used in one-show tags
- s - group manager's secret element in group signature scheme
- s_i - each player's secret element in group signature scheme ($1 \leq i \leq n$)
- S_i - each player's private input set ($1 \leq i \leq n$)
- $S = \bigcup_{i=1}^{n} S_i$
- t - number of values collected to estimate the number of distinct elements in a set
- T - number of filters
- u - estimated set size
- U_i - the tags collected during a t-collection or t-minimum value aggregation protocol by player i ($1 \leq i \leq n$)
- v - a one-show value
- $[w_{i,q,j}]_{j \in [b]}$ - local filter q ($1 \leq q \leq T$) for player i ($1 \leq i \leq n$)
- $[x_{q,j}]_{j \in [b]}$ - global filter q ($1 \leq q \leq T$)
- y_i - the root of a Merkle hash tree, a commitment to the input set S_i of player i ($1 \leq i \leq n$)
- z_0, z_1, z_2 - elements of the group signature public key for one-show tags
- Z_n - the ring of integers modulo n
- α - number of separate approximate distinct element counting estimations that must be combined to obtain the desired confidence probability δ_1
- β - secret value for ZK proof in one-show tags scheme
- γ - "gap" factor in approximate distinct element counting
- δ_1 - confidence probability for the approximate distinct element counting algorithm
- δ_+ - maximum probability of false positive
- δ_- - maximum probability of false negative
- ϵ - allowed error bound for the approximate distinct element counting algorithm is between $(1 - \epsilon)$ times the actual value and $(1 + \epsilon)$ times the actual value
- κ - security parameter for modified unpredictable-value group signature scheme
- λ - number of malicious players
- μ - secret value for ZK proof in one-show tags scheme
- η - part of public key for one-show scheme $\eta = r_1 r_2$
- ϕ - secret value for ZK proof in one-show tags scheme
- φ - maximum number of duplicates of each element allowed in a private input multiset (see Section 5.5)
- ρ - number of randomly chosen players chosen to receive anonymous messages in HotItem-ID and HotItem-Pub, so that, with high probability, at least one message reaches an honest player
- τ - threshold value for top-k protocol extension to HotItem-ID (see Section 5.5)
- σ - seed for pseudo-random number generator
- σ_ℓ - the ℓth member of Z_ν output by a pseudo-random number generator on input σ
- $\Phi(f_a)$ - expected size of indistinguishable set, if element a appears in f_a players' private inputs
- ψ - average number of neighbors per node

B.2 Detailed Analysis

B.2.1 Correctness

In this section, we analyze our HotItem-ID protocol to ensure that, given certain choices of the parameters b, T, our protocol correctly identifies hot items with high probability. In analyzing this protocol, we must consider both false positives (in which cold items are identified as hot), and false negatives. Errors may be caused by inaccurate approximate distinct element counting, a badly constructed filter, or a combination of the two.

Recall that $h_1, \ldots, h_T : \{0,1\}^\kappa \to \{1, \ldots, b\}$ are polynomial-wise independent hash function with uniformly distributed output, such as cryptographic hash functions, and that $\mathcal{H}$ is a collision-resistant cryptographic hash function [40].

Error from Approximate Distinct-Element Counting. To begin, we state a simple lemma that shows the t-collection and t-minimum value aggregation protocols (Figures 5.3 and 5.4) obtain the information needed for approximating the number of 'hits' to any particular bucket.

Lemma 32. *Each participant in the t-collection protocol of Figure 5.3 collects t 'small-value' one-show tags created by distinct players, if such tags exist.*

Each participant in the t-minimum value aggregation protocol of Figure 5.4 obtains the t one-show tags created by distinct players with the smallest hash values.

Proof. The proof of this theorem relies on the proof of information distribution in [46], which uses a closely related information distribution mechanism. We can thus observe that all information held by connected honest players is distributed to *all* connected players unless it is specifically filtered by an honest player. In the t-minimum value aggregation protocol, the only tags filtered are those which do not appear in the final result; that is, they are not one of the t one-show tags with smallest hash value. Note also that the one-show value of a tag is unique per player, and that each player cannot construct tags with different, valid one-show values with overwhelming probability. Thus, in the t-minimum value aggregation protocol, all players obtain the t one-show tags with smallest values created by distinct players.

We may reason similarly about our t-collection protocol. Each player only filters tags (by terminating his involvement in the protocol) if he has collected t tags and sent them on to all neighbors. Thus, the player has ensured that if t small hash-value tags exist, each of his neighbors collect them as well. By induction, all connected honest players thus collect t small-value tags created by distinct players, if such tags exist. □

As detailed in [4], if $\mathcal{H}(v)$ is the tth smallest hash value in a set S, where $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\kappa$, then the estimate of $|S|$ is $\frac{t2^\kappa}{\mathcal{H}(v)}$. By computing the median of $O\left(\lg \frac{1}{\delta_1}\right)$ such estimates, with computationally independent hash functions, we obtain $\overline{|S|}$, an (ϵ, δ_1)-approximation of $|S|$:

Lemma 33. *This algorithm, for any $\epsilon, \delta_1 > 0$, (ϵ, δ_1)-approximates $|S|$. That is,*

$$Pr\left[\overline{|S|} > (1+\epsilon)|S| \vee \overline{|S|} < (1-\epsilon)|S|\right] \leq \delta_1$$

Proof. Proof of this theorem is given in [4]. □

As this is an (ϵ, δ_1)-approximation algorithm, we may note that we are concerned with elements that appear fewer than $\frac{k}{1+\epsilon}$ times, when computing false positives, and elements that appear at least $\frac{k}{1-\epsilon}$ times, when computing false negatives. Based on this algorithm, we may conclude that, if there exist at least t elements with hash values at most $\frac{t2^\kappa}{k}$, our estimate of $|S|$ is at least k:

Corollary 34. *If there exist t values $v_1, \ldots, v_t \in S$ such that $\forall_{\ell \in [t]}\ \mathcal{H}(v_\ell) \leq \frac{t2^\kappa}{k}$, then the approximate distinct element counting algorithm of [4] will estimate that $|S| \geq k$.*

Proof. Let $w = \max_{\ell \in [t]}\{\mathcal{H}(v_\ell)\}$. The approximation algorithm specifies that $\overline{|S|} = \frac{t2^\kappa}{w}$. As $w \leq \frac{t2^\kappa}{k}$, then $\overline{|S|} \geq k$. □

Error from Filters. We now calculate the probability that an element a, which appears in $f_a < k'$ players' private input sets, is identified as a k'-threshold hot item. We set $k' := \frac{k}{1+\epsilon}$, so as to account for the allowed inaccuracy in the approximate counting algorithm.

As illustrated in Figure B.1, we may bound the probability that a is erroneously identified as k'-hot by **one** filter j' $(1 \leq j' \leq T)$ by determining the maximum number of filter buckets that were hit by $k' - f_a$ distinct players[1]. If a bucket j $(1 \leq j \leq b)$ was hit by $k' - f_a$ players who do not hold a, then if $h_{j'}(a) = j$, then a will be identified by filter j' as a k'-threshold hot item. As malicious players may claim to hit all buckets, a minimum of $k' - f_a - \lambda$ honest players must hit any given bucket for it to cause any possibility of error.

Each honest player has a maximum of m items in their private input set. Thus, using every element of each players' private input set, each group of $k' - f_a - \lambda$ honest players may hit at most m buckets a sufficient number of times to introduce any danger of error[2]. There are $n - f_a - \lambda$ honest players who do not hold a as an element of their private input set, and thus $\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor$ groups of players that can hit m buckets per group enough times to allow danger of an error. Note that any group of fewer than $k' - f_a - \lambda$ players cannot hit any particular bucket a sufficient number of times to cause a total of $k' - f_a$ hits; each player may only hit any particular bucket at most once.

Thus, at most $m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor$ buckets of each filter can be 'unsafe'; if a is not mapped to one of those buckets, then there is no possibility of error from the malfunctioning of the filter. As $h_{j'}(a)$ is distributed computationally indistinguishably from uniformly over $[b]$, we may thus bound the probability that bucket $h_{j'}(a)$ is erroneously designated as 'hot':

$$\Pr\big[a \text{ is identified as } k'\text{-hot by one filter}\big] \quad \leq \quad \frac{m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda} \rfloor}{b}$$

Combined Error. We may now consider the two sources of error together. There are two possible error types: false positives (in which cold items are identified as hot), and false negatives (in which hot items are not identified as hot).

Theorem 26. *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the*

[1] Note that the f_a signatures related to the element a will raise the total number of signatures to k'; if there are at least k' signatures, error in the approximate counting algorithm may cause a to be identified as a k-hot item.

[2] Note that, with high probability, more that one element of an honest players' private input set will hash to the same bucket, and thus this is a conservative analysis.

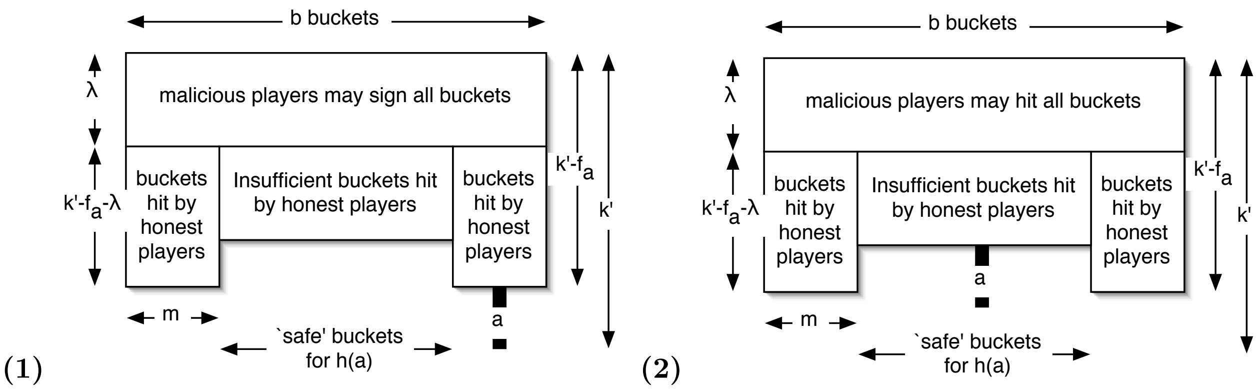

b buckets
λ
malicious players may sign all buckets
k'-f_a-λ
buckets hit by honest players
Insufficient buckets hit by honest players
buckets hit by honest players
k'-f_a
k'
m
`safe' buckets for h(a)
a
(1)
b buckets
λ
malicious players may hit all buckets
k'-f_a-λ
buckets hit by honest players
Insufficient buckets hit by honest players
buckets hit by honest players
k'-f_a
k'
a
m
`safe' buckets for h(a)
(2)

Figure B.1: Buckets that are unsafe for $h_{j'}(a)$ $(1 \leq j' \leq T)$ are those that have been marked as hit by a sufficient number of players to allow the possibility that a might be erroneously identified as a k'-threshold hot item. In **(1)**, a is mapped to a sufficiently full bucket, causing a to be erroneously identified as a k'-hot item. In **(2)**, a is mapped to a safe bucket.

following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, *$b$ and T are chosen to minimize $b \times T$, and at the same time, satisfy* $\left(\frac{m\lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda}\rfloor}{b} + \frac{\delta_-}{T}\right)^T < \delta_+$.

In the HotItem-ID *protocol, with probability at least $1-\delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1-\delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

Proof. The probability that an element a, which appears in $f_a < \frac{k}{1-\epsilon}$ players' private input sets, is not identified as a k-threshold hot item can be bounded as follows (recall that we have set $k' = \frac{k}{1+\epsilon}$, to account for the allowed tolerance in approximate counting):

$$\begin{aligned}
\Pr[a \text{ is identified as } k\text{-hot}] &\leq \Pr[\text{in all filters, element } a \text{ is identified as } k'\text{-hot} \vee \\
&\quad\ \text{set of size} < k' \text{ approximated as } \geq k] \\
&= \prod_{j'=1}^{T} \Pr[\text{in filter } j', \text{ element } a \text{ is identified as } k'\text{-hot} \vee \\
&\quad\ \text{set of size} < k' = \frac{k}{1+\epsilon} \text{ approximated as } \geq k] \\
&\leq \prod_{j'=1}^{T} \left(\frac{m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda}\rfloor}{b} + \delta_1\right) \\
&= \left(\frac{m\lfloor \frac{n-f_a-\lambda}{k'-f_a-\lambda}\rfloor}{b} + \delta_1\right)^T \\
&= \left(\frac{m\lfloor \frac{n-f_a-\lambda}{\frac{k}{1+\epsilon}-f_a-\lambda}\rfloor}{b} + \delta_1\right)^T
\end{aligned}$$

If an element a, which appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets, is not identified as a

k-threshold hot item, it is due to error in the set-counting approximation. When the number of hits for every bucket in a filter is counted exactly, there can be no false negatives. Thus, we may bound the probability of a false negative as follows:

$$\begin{aligned}
\Pr\left[a \text{ is not identified as } k\text{-hot}\right] &= \Pr\left[\text{in at least one filter, a set of size } \geq \frac{k}{1-\epsilon} \text{ approximated as } < k\right] \\
&\leq \sum_{j'=1}^{T} \Pr\left[\text{in filter } j'\text{, a set of size } \geq \frac{k}{1-\epsilon} \text{ approximated as } < k\right] \\
&\leq \sum_{j'=1}^{T} \delta_1 \\
&= \delta_1 T
\end{aligned}$$

□

Choice of Constants. Given our analysis, we may outline how to choose the constants δ_1, t, b, T based on the parameters $\epsilon, \delta_-, \lambda, n, m, \delta_d, \beta$.

δ_1. Recall that the probability of a false negative, for an element a which appears in at least $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets, is required to be at most δ_-. We may then choose δ_1 as follows:

$$\begin{aligned}
\Pr\left[a \text{ is not identified as } k\text{-hot}\right] &= \delta_- \\
&\leq \delta_1 T \\
\delta_1 &:= \frac{\delta_-}{T}
\end{aligned}$$

b, T. Given this assignment of δ_1, we may simplify our bound on the probability of a false positive on an element a, which appears in $\beta k < \frac{k}{1+\epsilon}$ players' private input sets, as follows:

$$\begin{aligned}
\Pr\left[a \text{ is identified as } k\text{-hot}\right] &\leq \left(\frac{m\left\lfloor\frac{n-f_a-\lambda}{\frac{k}{1+\epsilon}-f_a-\lambda}\right\rfloor}{b} + \delta_1\right)^T \\
&\leq \left(\frac{m\left\lfloor\frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda}\right\rfloor}{b} + \frac{\delta_-}{T}\right)^T
\end{aligned}$$

We choose b, T so as to minimize $b \times T$, while satisfying the above constraint.

t, α. In the approximate distinct element counting algorithm in [4], $t := \lceil\frac{96}{\epsilon^2}\rceil$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$. In practice, one may safely choose to retain $t < \lceil\frac{96}{\epsilon^2}\rceil$ smallest values per parallel execution, and run only one parallel execution, while retaining a confidence bound of δ_1. We found that $t := 25$ was sufficient.

Note that, when running very small examples, or with very high accuracy requirements, one may obtain an assignment for t that is $\geq k$. In this case, simply set $t := k$, and note that $\epsilon, \delta_1 = 0$; each player is now collecting a sufficient number of signatures so as to determine, without error, whether any particular bucket was hit by at least k distinct players.

ρ. The choice of ρ must be based on the choice of anonymous message system. We provide here an analysis based upon the scheme of [42], in which each node, upon receipt of a message to be anonymously routed, sends it to a random neighbor with probability $p_f > .5$, and to its intended destination with probability $1 - p_f$. Note that the probability that a message will, somewhere along its anonymously routed path, encounter a malicious node is at most $\frac{\lambda}{n} + \frac{\lambda}{n} p_f + \frac{\lambda}{n} p_f^2 + \cdots \leq \frac{\lambda}{n}$. As we cannot ensure that a message is delivered if it encounters a malicious node, we wish to choose ρ such that at least one message will be delivered to its destination, with probability at least δ_d. Thus, $\rho := O\left(\lg \frac{1}{\delta_d}\right)$.

B.2.2 Owner Privacy

Theorem 27. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the* HotItem-ID *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$).*

Proof. The proof of this theorem is straightforward. □

Theorem 29. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Correlated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$, for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Proof. The proof of this theorem is straightforward. □

Theorem 30. *Assume that one-show tags are unlinkable and that the anonymous communication system is secure such that no coalition of adversaries can distinguish which honest player sent any given anonymous message with probability more than negligibly different from a random guess. In the Uncorrelated Owner-Private* HotItem-Pub *protocol, for any element a, no coalition of at most λ malicious players can gain more than a negligible advantage in determining if $a \in S_i$ for any given honest player i ($1 \leq i \leq n$), assuming that the adversary is given the set of hot items P, and the frequency of each hot item.*

Additionally, given two elements $a, a' \in P$, no coalition of at most λ malicious players can gain more than a negligable advantage in determining if there exists a honest player i ($1 \leq i \leq n$) such that $a, a' \in S_i$, assuming that the adversary is given the set of hot items P, and the frequency of each hot item.

Proof. The proof of this theorem is straightforward. □

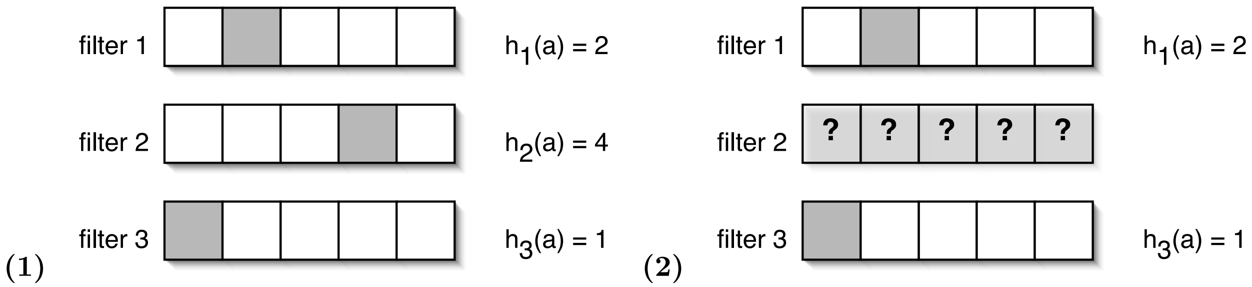


Figure B.2: Players collect one-show tags for each filter bucket during the HotItem-ID protocol. Given a complete set of filters for some element, one may still not determine which element produced the filters. However, each tag has a probability of $1-\frac{t}{k}-\frac{1}{2^\kappa}$ of having too high a value for the t-collection phase, and thus of being hidden. When all tags for an element in a specific filter are removed, as in **(2)**, an even larger number of elements could have produced the observed filters.

B.2.3 Data Privacy

Theorem 28. *In the* HotItem-ID *protocol, each element a, which appears in f_a distinct players' private input sets, has an indistinguishable set of expected size* $\Phi(f_a) = \sum_{\ell=1}^{T}\binom{T}{\ell}\left(1-\frac{t}{k}\right)^{f_a(T-\ell)}\left(1-\left(1-\frac{t}{k}\right)^{f_a}\right)^{\ell}\frac{|M|}{b^\ell}$.

Proof. Let M be the domain of possible private input set elements, such that $\forall_{i\in[n]}\ S_i \subseteq M$. Given knowledge of $h_1(a),\ldots,h_T(a)$, we may infer that approximately $\frac{|M|}{b^T}$ possible elements $a \in M$ could have produced that filter pattern, as there are b^T total possible filter patterns caused by one element and $h_1,\ldots,h_T$ are cryptographically secure hash functions. As illustrated in Figure B.2, if information about one or more filters has been elided, a correspondingly larger number of elements could have produced that filter pattern. We may use Bernoulli trials to calculate the expected size of the indistinguishable set for element a, which appears in f_a players' private input sets:

$$\begin{aligned}
\mathrm{E}\left[\text{size of indis. set}\right] &= \sum_{\ell=1}^{T}\Pr\left[\ell\text{ filters are elided}\right]\frac{|M|}{b^\ell} \\
&= \sum_{\ell=1}^{T}\binom{T}{\ell}\left(1-\frac{t}{k}\right)^{f_a(T-\ell)}\left(1-\left(1-\frac{t}{k}\right)^{f_a}\right)^{\ell}\frac{|M|}{b^\ell}
\end{aligned}$$

We graph in Figure 5.7 the expected proportion of the total domain M of the indistinguishable set, as f_a increases. Note that if a appears in only a few players' private input sets, a very large proportion of the domain is indistinguishable from a. As f_a approaches $\frac{k}{t}$, less and less of the domain is indistinguishable; this character ensures that truly rare elements are highly protected. As t is a constant, independent of n, while k will often grow with the size of the network, we see that protection for rare elements generally increases as the network increases in size.

□

B.2.4 Analysis for Bloom Filters

Theorem 31: *Given the false positive rate δ_+ and the false negative rate δ_-, error bounds ϵ and β, the upper limit of the number of malicious participants λ. Let b, t, T, ρ be chosen as the following: $t := \lceil \frac{96}{\epsilon^2} \rceil$, $\rho := O\left(\lg \frac{2}{\delta_-}\right)$, $\alpha := O\left(\lg \frac{2}{\delta_-}\right)$, b and T are chosen to minimize $b \times T$, and at the same time, satisfy $\left(\frac{mT\lfloor \frac{n-\beta k-\lambda}{\frac{k}{1+\epsilon}-\beta k-\lambda} \rfloor}{b} + \frac{\delta_-}{T}\right)^T < \delta_+$.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_+$, every element a that appears in $f_a < \beta k$ players' private input sets is not identified as a k-threshold hot item.*

In the HotItem-ID *protocol, with probability at least $1 - \delta_-$, every element a that appears in $f_a \geq \frac{k}{1-\epsilon}$ players' private input sets is identified as a k-threshold hot item.*

Proof. The proof of this theorem very closely follows the proof of Theorem 26. □

B.3 Details of One-Show Tags

In this section, we describe a one-show tag scheme obtained through the modification of the group signature scheme of Boneh and Shacham [7]. One one-show tag is lightweight, requiring only 1539 bits. Using the same techniques, similar constructions can be obtained from other group signature schemes.

A group signature scheme allows each member of group of players to sign messages on behalf of the group. Given these signatures, no player or coalition of players (except the trusted *group manager*) can distinguish the player that produced any signature, nor can they determine if two signatures were produced by the same group member.

In the Boneh/Shacham group signature scheme, the group public key is $pk = \{g_1, g_2, w\}$, where $g_1 \in G_1$, $g_2 \in G_2$, and $w = g_2^\gamma$ for $\gamma \leftarrow Z_p^*$. (p can be taken to be a 170-bit prime, and the elements of G_1, G_2 can be represented in 171 bits [7].) The trusted group manager holds the group secret key γ. Each user i $(1 \leq i \leq n)$ has a private key $s_i = \{A_i, x_i\}$, where $x_i \in Z_p^*$, $A_i \in G_1$. Using his private key, each player may sign a message, using a variant of the Fiat-Shamir heuristic [20], by proving knowledge of a pair $\{A_i, x_i\}$ such that $A_i^{x_i+\gamma} = g_1$.

We may modify this group signature scheme to include provably correct one-show values, making each signature a one-show tag. Each user i $(1 \leq i \leq n)$ may construct a one-show tag for bucket j of filter q $(1 \leq q \leq T, 1 \leq j \leq b)$ by: (1) signing the message $q \mid\mid j$ essentially as in the original signature scheme; (2) computing two additional values to enable the recipient to compute the one-show value.

Each user generates $g_{q,j} \in G_2$ by an agreed-on scheme; we discuss this element later in the section. We utilize the same bilinear mapping e as in the computation of the main signature, as well as the intermediate values v, α, r_α computed as intermediate values utilized in the main signature. User i computes these additional elements for a one-show tag:

$$\begin{aligned} T_3 &= e(v, g_{q,j})^{r_\alpha} \\ T_4 &= e(v, g_{q,j})^{\alpha} \end{aligned}$$

The sole change to the original signature scheme is that the challenge value c is computed as $c = H(pk, q \mid\mid j, r, T_1, T_2, T_3, T_4, R_1, R_2, R_3)$.

The recipient conducts all the validity checks specified in the Boneh/Shacham signature scheme, as well as the following additional check, derived from a proof of discrete logarithm equality [12]:

$$e(v, g_{q,j})^{s_\alpha} = T_4^c T_3$$

We define the one-show value as $e(A_i, g_{q,j})$; note that this value cannot be constructed by any player other than i and that player i can construct exactly one such value. To compute this value, the signature recipient computes:

$$\left(e(T_2, g_{q,j})^c e(v, g_{q,j})^{-s_\alpha} T_3\right)^{\frac{1}{c}}$$

The additional zero-knowledge proof required for the one-show tag construction is efficient, and thus our one-show tag construction and verification is nearly as efficient as the original group signature scheme. Note that these one-show tags are unlinkable, anonymous, and can be verified by all players who hold the group public key.

The parameter $g_{q,j}$, used to construct a one-show value associated with bucket j of filter q ($1 \leq q \leq T$, $1 \leq j \leq b$), can be efficiently constructed in a variety of ways such that no player knows its discrete logarithm. In this section, we briefly describe one such approach.

Let `PRNG` be a pseudo-random number generator with range G_2 [40]. Let H_{PRNG} be a hash function that takes bit strings as input and outputs data suitable for input into `PRNG`. Let $\sigma = H_{PRNG}(g \mid\mid q \mid\mid j)$. Let the ℓth element output from this `PRNG` on seed σ be denoted σ_ℓ. Each player calculates the element $\sigma_{\ell'}$, the first generator of G_2 in the sequence $\sigma_1, \sigma_2, \ldots$. Set $g_{q,j} = \sigma_{\ell'}$.